ANALISA PERFORMA SOKET *ITERATIVE SERVER* DAN *CONCURRENT FORK SERVER* PADA IPv4 DAN IPv6

Abid Abdul Ghofir[1], Lutfi Ali Muharom[2], EkoFajar Yanuarsa[3]
Jurusan Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Jember
abidabdulg@gmail.com

ABSTRAK

Client-Server memiliki beberapa alternative desain, terutama *Concurrent Server*. Pemilihan desain server yang benar membuat efisiensi dalam penggunaan waktu dan pengendalian proses Sebuah server memiliki kontrol proses lebih dari klien sebagai server harus merespon multi-permintaan dan multi-processing di waktu yang sama dari platform Client yang berbeda seperti IPv4 atau IPv6. Penelitian ini menganalisis kinerja IPv4 dan IPv6 pada Desain Concurrent Fork Server dan Concurrent Pre-Fork Server Percobaan untuk menganalisis CPU time termasuk kernel time untuk setiap server dilakukan pada soket TCP menggunakan beberapa teknik, termasuk menugaskan 20 dengan koneksi 20-100 koneksi berturut-turut untuk setiap klien pada setiap tes untuk setiap server.pada jaringan IPv4 dan Ipv6. Dari hasil percobaan pada *IPv4 Concurrent Fork Server* menggunakan CPU (42.446 %) dan lebih banya dari Iterative Server yang hanya (8.386%) sedangkan pada IPv6 Concurrent Fork Server menggunakan CPU (66.235%) daripada Concurrent pre fork server yang hanyamenggunakan (11.183%) perdetik dengan 100 koneksi setiap client. Dari hasi percobaan ini Iterative Server lebih handal dalam menangani permintaan dari client dibandingkandengan *Concurrent Fork Server.*

Kata kunci :*IPv4, IPv6, Iterative Server, Concurrent Fork Server*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dengan perkembangan teknologi internet yang semakin pesat menyebabkan *user* internet semakin bertambah banyak. Tetapi pertambahan user internet tidak diimbangi dengan jumlah alamat yang disediakan IPv4 (Samud, 2003). Sehingga IETF mengeluarkan standart protokol IP baru yang disebut IPng (*Internet Protokol Next Generations*) atau disebut juga IPv6. Alamat pada IPv4 pada dasarnya menggunakan pengalamatan berbasis 32-bit, yang mampu mengakomodasi jumlah pengalamatan sampai dengan 2^{32} atau skitar 4,294 x 10^9. IPv6 merupakan standar pengalamatan internet berbasis 128-bit, sehingga secara teoritis dapat mengalamati hingga 2^{128}=3,4 x 10^{38} *host* komputer di seluruh dunia. Dengan terbatasnya jumlah jumlah IP yang bisa digunakan dalam protokol versi 4 ini merupakan alasan utama digagasnya IPv6 untuk mengatasi kekurangan pada IPv4. Sepeti yang kita ketahui saat ini internet kebanyakan masih menggunakan IPv4. Dan baru beberapa yang mulai menggunakan. Tidak menutup kemungkinan bahwa kedepanya semuanya akan beralih menuju ke IPv6 walau perubahan tersebut dilakukan secara

bertahap sebagai penyesuaian (Gilang Ramadhan Paramayudha FT UI.2010).

IPv4 dan IPv6 memiliki struktur yang berbeda, misalnya format *header*. Beberapa bidang Format header IPv4 tidak lagi tersedia atau digantikan dalam header IPv6, seperti 6-bit DSCP *field* dan 2-bit ECN *field* menggantikan *historical 8-bit traffic class field*, panjang 16 bit *payload* tidak disertakan di IPv6, dll. Hal ini bertujuan untuk meningkatkan kecepatan data *forwarding* dan mengurangi *delay*.

Selain masalah pada IPv4 dan IPv6, desain *Server* juga penting dalam mengelola transfer data antara beberapa *dedicated Server* dan beberapa *client* terutama dalam kasus *sending* dan *requesting queries* dari beberapa *client* dan pada saat yang sama *client* meminta beberapa tanggapan juga. Client - *Server* memiliki beberapa alternatif desain, terutama iterative *Server* dan concurrent fork *Server*. Sebuah *Server* memiliki kontrol proses lebih dari klien sebagai *Server* harus menanggapi multi- query dan multi-processing di waktu yang sama dari platform klien yang berbeda seperti IPv4 atau IPv6 . Studi ini menganalisis kinerja IPv4 dan IPv6 di 2 desain *Server* yang berbeda , yaitu iterative *Server*, Concurrent Fork *Server*,. Percobaan untuk menganalisis CPU time termasuk kernel untuk setiap *Server* dilakukan pada soket TCP menggunakan beberapa teknik, itu termasuk menugaskan 20 klien dengan koneksi 50-100 koneksi berturut-turut untuk setiap klien pada setiap tes untuk setiap *Server* . Penelitian ini membandingkan, dibahas dan dianalisis alokasi waktu untuk setiap jenis *Server* tersebut untuk menanggapi permintaan dari klien.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat dirumuskan permasalahan yang akan diangkat dalam Tugas Akhir ini yaitu

a. Bagaimana Membandingkan kinerja dari dua model *Server Iterative Server, Concurrent Fork Server* pada Ipv4 dan Ipv6?.

b. Bagaimana analisa CPU *time* dan *kernel time* pada proses yang terjadi di*Server?*.

1.3 Batasan Masalah

Agar tidak menyimpang jauh dari permasalahan, maka Tugas Akhir ini mempunyai batasan masalah sebagai berikut :

a. Menetapkan 20 *client* dengan koneksi 50-100 koneksi *consecutive* untuk setiap *client* untuk setiap tes untuk setiap *Server*.

b. Menetapkan 20 *client* dengan 10 koneksi dan 16.000 *byte* untuk setiap *child.*

c. Menetapakan 5 *client* dan 100 koneksi untuk setiap *Server* dalam membandingkan kinerja IPv4 dan IPv6

d. *Server* dan client dijalankan dengan os linux.

e. Tidak membahas pengalamatan pada IPv4 dan IPv6 secara spesifik.

1.4 Tujuan Penulisan

Tujuan dari Tugas Akhir ini adalah untuk mengetahui kinerja CPU *time* dan *kernel time* dari dua desain *Server* yaitu *Iterative Server* dan *Concurrent Fork Server* pada IPv4 dan IPv6.

1.5 Manfaat Penulisan

Dari hasil penelitian ini diperoleh manfaat yaitu mengetahui performa Iterative Server dan Concurrent Fork Server pada IPv4 dan IPv6

II. Tinjauan Pustaka

2.1 *Iterative Server*

Iterative server melayani satu *client* pada suatu waktu. memproses permintaan *client* pertama sampai selesai baru kemudian melayani *client* berikutnya. Karena *iterative Server* hanya berfungsi dengan satu *client* pada suatu waktu, tidak ada kontrol proses yang dilakukan oleh *Server*. Ini berkaitan dengan *client* dengan mengalokasikan soket dan menempatkannya ke *listen mode* dan menetapkan jumlah maksimum koneksi. Setelah menerima permintaan sambungan masuk, itu akan membuat *socket* baru untuk sambungan. Kemudian memproses data yang masuk dan merumuskan respon serta mengirimkan data keluar di sambungan. Setelah selesai dengan *client* tertentu, *Iterative server* akan menutup sambungan dan mengalokasikan *socket* dan kembali ke modus 5. mendengarkan menunggu permintaan sambungan berikutnya. Sebagai karakteristiknya, ia menetapkan satu *socket* untuk setiap *client* tunggal (Teddy Mantoro**,** Media A. Ayu, Amir Borovac and Aqqiela Z. Z. Zay2012 IEEE).

2.1 *Concurrent Fork Server*

Concurent server adalah sebuah *server* yang mempunyai kemampuan melayani beberapa *client* pada suatu waktu. *Concurent-fork server* menggunakan fungsi *fork* dalam menangani banyak *client*. *Server* menciptakan 1 *child* untuk setiap *client* sehingga memiliki 1 *client* proses. Ketika *server receive* dan *accept* sambungan *client*, maka *server* akan membuat *child* (*copy*-an dari *server*) dan memungkinkan *child* menangani *client*. *fork* berfungsi membuat proses baru yang terpisah untuk setiap *client* tunggal. *Frok* berfungsi membuat proses baru yang terpisah dari setiap *client* tunggal, dan

membuat proses menjadi *parent* dan *child*. *Child* yang mengacu pada proses baru memiliki *frame* atau model yang sama (mengacu pada proses induk) (Teddy Mantoro**,** Media A. Ayu, Amir Borovac and Aqqiela Z. Z. Zay2012 IEEE).. Ketika *server* menerima dan menerima koneksi klien, fork tersebut akan mengkopy dirinya sendiri dan memungkinkan child menangani klien.

2.2 IPv4

Alamat IP versi 4 adalah sebuah jenis pengalamatan jaringan yang digunakan di dalam protocol jaringan TCP/IP yang menggunakan protokol IP versi 4.Panjang totalnya adalah 32-bit, dan secara teoritis dapat mengalamati hingga 4 miliar host computer atau lebih tepatnya 4.294.967.296 host di seluruh dunia, jumlah host tersebut didapatkan dari 256 (didapat kandari 8 bit) dipangkat 4(karena terdapat 4 oktet) sehingga nilai maksimal dari alamt IP versi 4 tersebut adalah 255.255.255.255 dimana nilai dihitung dari nol sehingga nilai host yang dapat ditampung adalah 256x256x256x256=4.294.967.296 host.

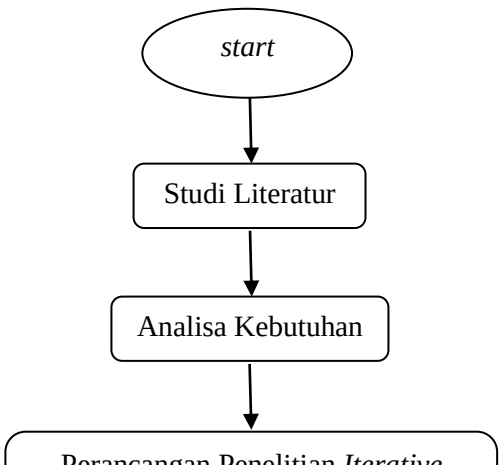
2.3 IPv6

Setiap komputer yang terhubung ke internet setidaknya haru smemiliki sebuah IP *address* pada setiap *interface*-nya dan IP *address* sendiri harus unik karena tidak boleh ada komputer / *server* /perangkat *network* lainnya menggunakan IP *address* yang sama di internet.

Dalam perkembangannya dunia internet sampai denga saat ini berkembang cukup drastic sehingga untuk pemakaian IPv4 dirasakurang memfasilitasi kebutuhan berinternet masyarakat dunia.Untuk itulah IETF (*Internet Engineering Task Force*) mendesain alamat IP baru yang disebut IPv6 (*internet Protocol Vesion*6). Protocol IPv6 dikembangkan untuk menangan imasalah banyaknya alamat yang tidak tertangani oleh IPv4, karena dunia internet tiap tahun berkembang sehingga perlu diganti dengan *Protocol* IPv6. IPv6 memilik ipanjang total 128-bit, dan secara teoritis dapat mengalamati hingga 2^{128}=3,4 x 10^{38}*host* komputer di seluruh dunia.

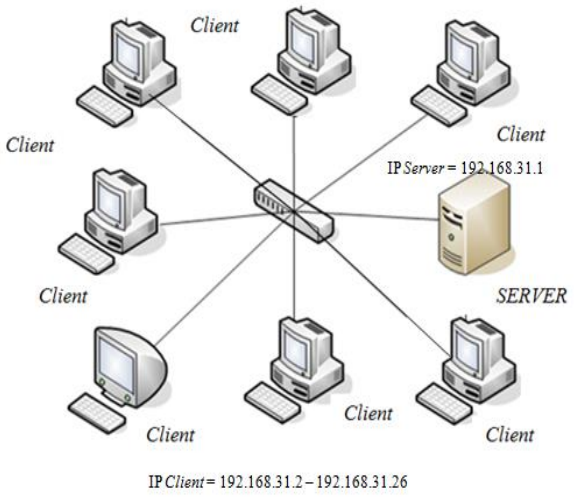
2.4 Konsep Penelitian

Berikut gambaran dari rancangan penelitian



III. HASIL DAN PEMBAHASAN

3.1 Konfigurasi IPv4



Pada konfigurasi ini computer *Server* diberi alamat IP 192.168.31.1/24 dan untuk computer *Client* yang pertama diberi alamat IP 192.168.31.2 sampai komputer yang ke 20 di beri IP 192.168.31.26 secara berurutan.

4.1.1 Performa *CPUTime* dan *Kernel Iterative Serve r*dan *Concurrent fork Server* pada IPv4

Pengujian performa *CPU Time* dan *KernelTime* dilakukan dengan bantuan tool aplikasi NMON yang dipasang di computer *Server*. Pengujian dilakukan dengan cara mengaktifkan tool NMON agar mencatat semua proses selama 1 menit setelah itu computer *Client* melakukan request koneksi kekomputer *Server*, dengan melakukan koneksi 20, 40, 60, 80, 100 koneksi dan 1000 byte setiap *Client*

a. CPU time

CPUTime adalah jumlah waktu yang dibutuhkan di dalam central processing unit (*CPU*) yang digunakan untuk pengolahan instruksi dari program komputer atau system operasi. Penggunaan *CPU* diperoleh dari hasil capture NMON di bawah ini adalah table dari 5 proses uji coba.

Tabel CPU time pada Ipv4

Tabel 4.1*CPUTime* pada IPv4

koneksi	***Iterative Server***	***Concurrent Fork Server***
20	2.137%	12.597 %
40	6.365%	21.004 %
60	6.870%	28.621 %
80	7.412%	35.397 %

100	8.386%	42.446 %

Dari ujicoba yang dilakukan bisa dilihat perbandingan penggunaan *CPU* dimana *Concurrent Fork Server* lebih banyak menggunakan *CPU* (42.446 %) dengan 100 koneksi per *Client* dibandingkan *Iterative Server* yang hanya menggunakan *CPU* (8.386 %) pada jaringan IPv4.

b. Kernel

Kernel adalah suatu perangkat lunak yang menjadi bagian utama dari sebuah sistem operasi.Tugasnya melayani bermacam program aplikasi untuk mengakses perangkat keras komputer secara aman.

KernelTime adalah jumlah waktu yang dibutuhkan dalam menjalankan beberapa peroses yang dibutuhkan oleh aplikasi

Tabel Kernel pada IPv4

koneksi	***Iterative Server***	***Concurrent Fork Server***
20	701.808	2.442,238
40	1.242,874	4.242,874
60	1.490,223	5.597,362
80	1.557.075	7.012,816
100	1.590.807	10.894,123

Dari ujicoba yang dilakukan bisa dilihat perbandingan penggunaan *Kernel* dimana *Concurrent Fork Server* lebih banyak menggunakan *Kernel* (10.894,123proses) dengan 100 koneksi per *Client* dibandingkan *Iterative Server* yang hanya menggunakan *Kernel* (1.590.807 proses) pada jaringan IPv4.

4.1.2 Performa Performa *CPUTime* dan *Kernel Iterative Server* dan *Concurrent fork Server* pada IPv6

Pengujian performa *CPU Time* dan *KernelTime* dilakukan dengan bantuan tool aplikasi NMON yang dipasang di computer *Server*. Pengujian dilakukan dengan cara mengaktifkan tool NMON agar mencatat semua proses selama 1 menit setelah itu computer *Client* melakukan request koneksi kekomputer *Server*, dengan melakukan koneksi 20, 40, 60, 80, 100 koneksi dan 1000 byte setiap *Client*

a. CPU time

Pengujian performa *CPUTime* dan *KernelTime* dilakukan dengan bantuan tool aplikasi NMON yang dipasang di komputer *Server*. Pengujian dilakukan dengan cara mengaktifkan tool NMON agar mencatat semua proses selama 1

menit setelah itu komputer *Client* melakukan request koneksi ke komputer *Server*, dengan melakukan koneksi 20, 40, 60, 80, 100 koneksi dan 1000 byte setiap *Client*

Tabel *CPU time* padaIPv6

koneksi	***Iterative Server***	***Concurrent Fork Server***
20	6.432 %	9.357 %
40	6.470 %	23.532 %
60	7.600 %	35.822 %
80	9.662 %	53.140 %
100	11.183 %	66.235 %

Dari ujicoba yang dilakukan bisa dilihat perbandingan penggunaan *CPU* dimana *Concurrent Fork Server* lebih banyak menggunakan *CPU* (66,235 %) dengan 100 koneksi per *Client* dibandingkan *Concurrent Pre-Fork Server* yang hanya menggunakan *CPU* (11.183 %) pada jaringan IPv6.

b. Kernel

Kernel adalah suatu perangkat lunak yang menjadi bagian utama dari sebuah sistem operasi.Tugasnya melayani bermacam program aplikasi untuk mengakses perangkat keras komputer secara aman.

KernelTime adalah jumlah waktu yang dibutuhkan dalam menjalankan beberapa pero ses yang dibutuhkan oleh aplikasi

Tabel Kernel pada IPv6

koneksi	***Iterative Server***	***Concurrent Fork Server***
20	1.111,453	1.704,520
40	1.603.707	4.536,770
60	1.704,520	7.233,632
80	1.745,997	10.894,123
100	2.165.940	13.339.745

Dari ujicoba yang dilakukan bisa dilihat perbandingan penggunaan *Kernel* dimana *Concurrent Fork Server* lebih banyak menggunakan *Kernel* (13.339.745 proses) dengan 100 koneksi per *Client* dibandingkan *Iterative Server* yang hanya menggunakan *Kernel* (2.165.940proses) pada jaringan IPv6.

IV. PENUTUP

Dari hasil ujicoba didapatkan hasil sebagai berikut :

1. Dari *Iterative Server* dimana proses yang dibangun untuk menangani setiap koneksi masuk setelah diminta, hasil penelitian menunnjukan bahwa butuh 8.386% per detik penggunaan CPU dalam menangani 100 koneksi per client untuk koneksi IPv4. sementara 11.183% per detik untuk koneksi IPv6. Sedangkan penggunaan kernel pada IPv4 membutuhkan 1.590.807 proses per detikuntuk 100 koneksi per client sementarapada IPv6 membutuhkan 2.165.940 proses per detik.
2. Dari Concurrent Fork Server dimana child server sudah siap sebelum

permintaan koneksi diminta, butuh 42.446 % per detik penggunaan CPU dalam menangani 100 koneksi per client untuk koneksi IPv4 sementara 66.235 % per detik untuk koneksi IPv6. Sedangkan penggunaan kernel pada IPv4 membutuhkan 10.894,123 proses per detik untuk 100 koneksi per client sementara pada IPv6 membutuhkan 13.339.745 proses per detik.

3. Berdasarkan hasil penelitian ini menunjukkan bahwa kinerja *Concurrent Fork Server* lebih berat dibandingkan *Iterative server* pada jaringan IPv4 maupun IPv6. Dan

DAFTAR PUSTAKA

[1] Anonim. BAB III METODOLOGI PENELITIAN. Di Download pada 17 April 2014. http://www.digilib.its.ac.id/public/ITS-Undergraduate-5121-4201100038-bab3.pdf.

[2] Grox.net.*IPv6calculator*. Diakses 24 Oktober 2014. http://grox.net/utils/ipv6.php.

[3] Indowebsiana.Com. *Apa itu Teknologi IPv6 dan Mengapa ini Sangat Penting.. ?*. Diakses 24Oktober 2014. http://www.indowebsia.com/shownews.php?news_id=74.

[4] Nmonvisualizer.github.io. NMONvisualizer diakses 23 desember 2014 http://nmonvisualizer.github.io/nmonvisualizer/index.html

[5] John, J., Minoli, D. Amoss.(2007). *Handbook of IPv4 to IPv6 Transition*. Auerbach Publications

[6] Rikih Gunawan. Pemrograman Socket dengan Pythondiakses 27 oktober 2014. http://www.unej.ac.id/files/pdf2/rikih-socket_python.pdf

[7] Syafrizal, Melwin. *TCP/IP*. Di *Download* pada 26 Oktober 2014. http://journal.amikom.ac.id/index.php/KIDA/article/view/4481/2175

[8] Teddy Mantoro**,** Media A. Ayu, Amir Borovac and Aqqiela Z. Z. Zay Department of Computer Science, KICT International Islamic University Malaysia, Kuala Lumpur, Malaysia, teddy@ieee.org

[9] Ti.unmuhjember.ac.id. PanduanTugasAkhir 2014 di *download*pada 24 Oktober 2014. http://ti.unmuhjember.ac.id/wp-content/uploads/2014/12 /Panduan_Proposal_20141.pdf

[10] Widiyansah, AA.(2013). *Perancangan Jaringan Laboratorium Komputer Universitas Muhammadiyah Jember Menggunakan Internet Protocol Version 6 (IPv6)*.(Skripsi). Jurusan Teknik Informatika Universitas Muhammadiyah Jember.