ISSN 2355-3286

Pengurangan Noise Sepeda Motor dan Mesin Diesel dari Sinyal Bicara dengan Algoritma Recursive Least Square

Hugeng[1], Endah Setyaningsih[2], Meirista Wulandari[2]
[1]Program Studi Sistem Komputer, Universitas Mutimedia Nusantara, Tangerang, Indonesia
hugeng@umn.ac.id
[2]Jurusan Teknik Elektro, Universitas Tarumanagara, Jakarta
endah.setyaningsih@tarumanagara.ac.id, meiristawulandari@hotmail.com

Diterima 1 Mei 2013
Disetujui 15 Mei 2013

Abstrak—Adanya bunyi kendaraan bermotor yang tercampur dengan suara seseorang yang sedang berbicara dapat mengganggu suatu sistem contohnya pada sistem *speech recognition* sehingga perintah terhadap sistem tersebut tak dapat dikerjakan. Ada beberapa cara yang dapat digunakan untuk mengatasi masalah gangguan noise yaitu salah satunya menggunakan filter adaptif dengan metode *Adaptive Noise Cancellation* (ANC). ANC menghilangkan *noise* yang tercampur dengan suatu sinyal berdasarkan noise referensi. ANC ini terdiri dari 2 bagian yaitu filter digital dan algoritma adaptif. Filter digital FIR dan algoritma adaptif RLS digunakan pada sistem ini. Pemfilteran menggunakan perangkat lunak Matlab secara simulasi dan hasil filter berupa sinyal estimasi. Keberhasilan sistem pengurangan noise ini dapat dilihat berdasarkan parameter *Mean Square Error* (MSE). Hasil parameter yang didapat menunjukkan bahwa sistem ini bisa mengurangi noise sepeda motor dan mesin diesel yang tercampur dengan sinyal bicara walau pun nilai MSE yang dihasilkan cukup besar.

Keywords—*Adaptive Filter, ANC, RLS*

I. Pendahuluan

Satu kesatuan yang membawa suatu informasi disebut sinyal. Sinyal yang ditransmisikan dari suatu sumber ke tempat tujuan biasanya terdiri dari dua bagian, yaitu bagian yang diinginkan, yang disebut sinyal asli dan bagian yang tidak diinginkan, yang sering disebut *noise*. Ada berbagai macam *noise* dalam kehidupan sehari-hari, salah satunya adalah *noise* akustik pada sistem bunyi. *Noise* akustik adalah bunyi yang berasal dari sumber lain di sekitar sistem tersebut, seperti bunyi dering telepon atau bunyi deru kendaraan yang melintas. Ada beberapa cara yang dapat digunakan untuk mengatasi masalah *noise* akustik seperti penggunaan bahan-bahan yang kedap suara sebagai insulasi terhadap *noise*. Pemakaian bahan-bahan kedap suara tersebut dapat ditemui pada studio-studio musik. Ada cara lain untuk mengatasi *noise* yaitu menggunakan filter digital yang merupakan salah satu aplikasi dari sistem pemrosesan sinyal digital. Penggunaan filter digital bisa digunakan untuk mengatasi masalah *noise* yang dapat berubah sepanjang waktu, seperti curah hujan, bunyi kendaraan bermotor, karena kemampuannya yang adaptif melalui struktur operasi matematika dan algoritma pemfilteran *noise*. Filter digital dengan kemampuan adaptif ini disebut dengan filter adaptif. Filter adaptif untuk sistem pengurang *noise* disebut dengan *Adaptive Noise Cancellation* (ANC). ANC ini mempunyai 2 bagian yaitu filter digital dan algoritma adaptif. *Finite Impulse Response* (FIR) sebagai filter digital dan algoritma *Recursive Least Square* (RLS) sebagai algoritma adaptif diuji untuk mengurangi *noise* sepeda motor dan mesin diesel yang tercampur dalam suatu sinyal bicara.

II. Adaptive Noise Cancellation

Dalam sistem ANC terdapat dua buah *input* yaitu sinyal yang tercampur *noise*, **d**(*n*), dan sinyal referensi *noise*, **x**(*n*). Kedua sinyal ini dimasukkan secara bersamaan ke dalam ANC dan kemudian diproses oleh filter digital untuk menghasilkan sinyal *noise* estimasi, **y**(*n*), yaitu sinyal yang mendekati sinyal referensi *noise*, **x**(n). *Input* sinyal+*noise* kemudian dikurangi dengan sinyal *noise* estimasi, **y**(*n*), sehingga dihasilkan sinyal estimasi yang mirip dengan sinyal informasi tanpa *noise*. Persamaan untuk mendapatkan sinyal estimasi **e**(*n*) adalah sebagai berikut:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n) = \mathbf{S}(n) + \mathbf{N}(n) - \mathbf{y}(n) \quad (1)$$

dimana **e**(*n*) = sinyal estimasi

d(*n*) = sinyal tercampur *noise*

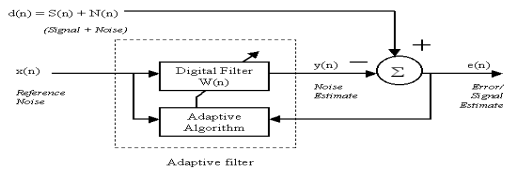
S(*n*) = sinyal informasi

N(*n*) = sinyal *noise*

y(*n*) = sinyal *noise* estimasi.

Sinyal estimasi yang dihasilkan oleh ANC ini kemudian dibandingkan dengan sinyal asli tanpa *noise*

agar dapat diketahui seberapa baiknya ANC ini dapat mengurangi *noise*.



Gambar 1. Diagram Blok ANC[6]

III. Recursive Least Square

Algoritma adaptif *Recursive Least Square* (RLS) dapat dijelaskan oleh persamaan berikut:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mathbf{k}(n)\, e(n) \quad (2)$$

$$e(n) = d(n) - \mathbf{W}^T(n)\, \mathbf{x}(n) \quad (3)$$

$$\mathbf{k}(n) = \frac{\mathbf{z}(n)}{\mathbf{x}^T(n)\mathbf{z}(n)+1} \quad (4)$$

$$\mathbf{z}(n) = \lambda^{-1}\, \mathbf{Q}(n-1)\, \mathbf{x}(n) \quad (5)$$

$$Q(n) = \lambda^{-1}\, \mathbf{Q}(n-1) - \mathbf{k}(n)\, \mathbf{z}^T(n) \quad (6)$$

dengan inisialisasi matriks $\mathbf{Q}(n)$ adalah

$$\mathbf{Q}(0) = \delta^{-1}\mathbf{I} \quad (7)$$

dimana $\mathbf{I}$ = matriks identitas dan δ = konstanta yang diatur sekecil mungkin dan nilai awal dari $\mathbf{W}(n)$ adalah $\mathbf{W}(0) = [00\ldots0]$. Kedua nilai awal ini disebut *soft-constrained initialization*.

Forgetting factor, λ, pada algoritma RLS, merupakan faktor yang secara eksponensial memberikan bobot yang lebih kecil kepada sampel-sampel *error* yang lebih lama dengan rentang nilai $0 < \lambda \leq 1$. *Forgetting factor* ini berguna untuk memastikan bahwa semakin jauh jarak data yang sebelumnya dengan data yang sedang diamati maka data tersebut semakin "dilupakan". Hal ini dilakukan demi mendukung kemungkinan variasi statistik data yang sedang diamati ketika filter beroperasi pada lingkungan yang tidak tetap.

IV. Parameter-parameter Kinerja Pengurang Noise

Parameter-parameter yang digunakan untuk mengetahui kinerja pengurang *noise* yang dirancang adalah *Signal-to-Noise Ratio* (SNR), *% crest factor* dan *Mean-Square Error* (MSE).

A. Signal-to-Noise Ratio

Daya sinyal merupakan energi yang terdapat dalam suatu sinyal dalam suatu satuan waktu. Daya sinyal dari suatu sinyal $\mathbf{x}(n)$ dapat dihitung dengan persamaan:

$$P(dB) = 10 \log\left[\frac{1}{N}\sum_{N=0}^{N-1} x^2(n)\right] \quad (8)$$

dimana N adalah jumlah sampel dalam sinyal $\mathbf{x}(n)$.

SNR adalah perbandingan antara daya sinyal asli terhadap daya sinyal *noise*. SNR yang semakin besar menunjukkan bahwa kualitas system pengurang *noise* adaptif semakin baik. SNR dapat dihitung dengan persamaan:

$$SNR\ (dB) = 10 \log\left[\frac{P_1}{|P_2 - P_1|}\right]\left[\frac{P_1}{|P_2 - P_1|}\right] \quad (9)$$

dimana P_1 = daya sinyal asli, $|P_2 - P_1|$= daya *noise* yang tersisa dan P_2 = daya sinyal estimasi.

B. Mean Square Error

Perbedaan antara sinyal estimasi dan sinyal asli menyebabkan terjadinya *error*. Kualitas suatu filter adaptif dapat dikatakan baik, jika MSE yang dihasilkan mendekati 0. Nilai MSE dapat dihitung dengan persamaan:

$$MSE = \frac{1}{N}\sum_{N=0}^{N-1}[u(n) - e(n)]^2 \quad (10)$$

dengan u(n) adalah sinyal asli, e(n) adalah sinyal estimasi dan N adalah panjang sampel sinyal.

C. Crest Factor

Crest Factor merupakan perbandingan nilai tertinggi suatu sinyal dengan nilai *root mean square* sinyal tersebut dengan faktor pengali akar kuadrat dari jumlah sampling sinyal. *Crest factor* dapat dihitung dengan persamaan:

$$cf(x) = \frac{\max_{n=0,1,\ldots,N-1}|x(n)|}{\sqrt{\frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2}} = \sqrt{N}\,\frac{\|x(n)\|_\infty}{\|x(n)\|_2} \quad (11)$$

Persentase perbedaan *crest factor* dari sinyal uji dan sinyal estimasi dapat dihitung pada persamaan:

$$\Delta C\ (\%) = 100\% \times |C_u - C_e| / C_u \quad (12)$$

dengan C_u adalah *crest factor* sinyal asli dan C_e adalah *crest factor* sinyal estimasi.

V. Pengujian dan Analisis

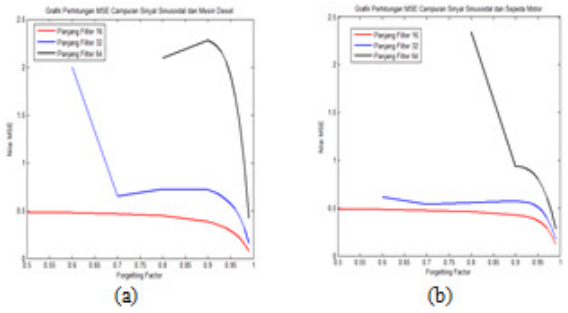
Sistem pengurang *noise* ini diuji dengan parameter kinerja yaitu MSE, SNR dan *crest factor*. Pengujian dilakukan terhadap tiga macam sinyal asli, yang kemudian di sini disebut sebagai sinyal-sinyal uji. Kinerja pengurang noise diobservasi dengan mengubah-ubah variabel *forgetting factor* (mulai dari 0,5 sampai dengan 0,99) dan mengubah-ubah panjang filter adaptif RLS. Ketiga sinyal uji dalam sistem pengurang *noise* ini adalah sinyal uji sinusoidal, sinyal uji segitiga dan sinyal suara bicara yang masing-masing berdurasi 3 detik atau 132.300 sampel dengan frekuensi sampling 44.100 Hz. Ketiga sinyal uji kemudian dicampur masing-masing dengan *noise*

berupa bunyi sepeda motor dan bunyi mesin diesel.

Sinyal uji sinusoidal dan sinyal uji segitiga dibangkitkan menggunakan program Matlab sedangkan bunyi sepeda motor, bunyi mesin diesel dan sinyal bicara didapatkan melalui perekaman. Hasil perekaman dikonversi menjadi data digital melalui program Matlab sehingga dapat ditampilkan dalam bentuk grafik dan direproduksi ulang agar dapat didengar melalui *speaker*. *Noise* sepeda motor yang digunakan adalah bunyi dari motor dengan jenis mesin 4 langkah dan kapasitas silinder 110 cc. *Noise* mesin diesel yang digunakan adalah bunyi mesin diesel dengan daya *output* 11 HP/3600 rpm.

A. Pengujian dengan Sinyal Uji Sinusoidal

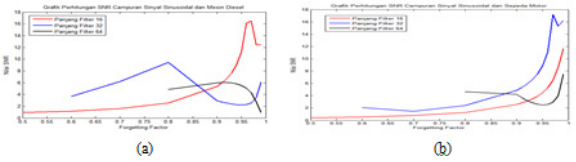
Sinyal uji sinusoidal dibangkitkan menggunakan program Matlab dengan frekuensi 997 Hz, sebagai perwakilan dari rentang frekuensi suara. Sinyal ini dicampur secara aditif dengan *noise* kemudian diproses oleh filter adaptif RLS untuk menghasilkan sinyal estimasi yang bebas dari *noise*. Hasil sinyal estimasi dianalisis dengan parameter MSE, SNR dan *crest factor*. Nilai-nilai MSE, SNR dan *crest factor* yang diperoleh dari hasil pengujian dapat dilihat dalam bentuk grafik untuk keperluan analisis lebih lanjut.



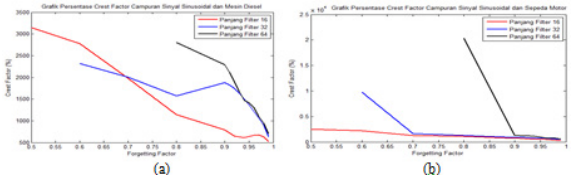
Gambar 2. Nilai-nilai MSE dari Pengurangan *Noise* pada Campuran Sinyal Uji Sinusoidal dengan (a) *Noise* Mesin Diesel dan (b) *Noise* Sepeda Motor

Seperti terlihat dari Gambar 2, hasil pengujian dengan sinyal uji sinusoidal menunjukkan bahwa nilai MSE yang paling kecil didapatkan dengan panjang filter 16 dan *forgetting factor* 0,99, untuk kedua macam *noise*.

Nilai SNR terbesar (16,53 dB) dihasilkan dengan panjang filter 16 dan *forgetting factor* 0,97 untuk campuran sinyal uji sinusoidal dan *noise* mesin diesel, seperti ditunjukkan Gambar 3(a). Tetapi untuk campuran sinyal uji sinusoidal dan *noise* sepeda motor dihasilkan nilai SNR terbesar (17,17 dB) pada panjang filter 32 dan *forgetting factor* 0,97, seperti terlihat dari Gambar 3(b).



Gambar 3. Nilai-nilai SNR dari Pengurangan *Noise* pada Campuran Sinyal Uji Sinusoidal dengan (a) *Noise* Mesin Diesel dan (b) *Noise* Sepeda Motor



Gambar 4. Nilai-nilai Persentase *Crest Factor* dari Pengurangan *Noise* pada Campuran Sinyal Uji Sinusoidal dengan (a) *Noise* Mesin Diesel dan (b) *Noise* Sepeda Motor

Untuk kedua macam *noise* yang dicampurkan dengan sinyal uji sinusoidal, diperoleh bahwa persentase *crest factor*, ΔC, cenderung bertambah kecil seiring dengan meningkatnya *forgetting factor*, λ, yang digunakan. Nilai-nilai persentase *crest factor* yang terkecil terjadi pada filter adaptif dengan panjang 16, seperti terlihat dari Gambar 4.

B. Pengujian dengan Sinyal Uji Segitiga

Sinyal uji segitiga juga dibangkitkan menggunakan program Matlab dan dicampur secara aditif dengan *noise* kemudian diproses oleh filter adaptif RLS untuk menghasilkan sinyal estimasi yang bebas dari *noise*. Hasil sinyal estimasi dianalisis dengan parameter MSE, SNR dan persentase *crest factor*. Hasil-hasil perhitungan nilai-nilai MSE, SNR dan persentase *crest factor*, yang diperoleh dari pengujian, dapat dilihat dalam bentuk grafik. Hasil-hasil ini hampir sama dengan hasil-hasil pengujian dengan sinyal uji sinusoidal.

Hasil pengujian dengan sinyal uji segitiga menunjukkan bahwa nilai-nilai MSE yang dihasilkan dengan panjang filter 16 dan berbagai macam nilai *forgetting factor,* lebih kecil dibanding nilai-nilai MSE dengan panjang filter lainnya. Semakin panjang suatu filter maka semakin besar pula nilai MSE yang dihasilkan untuk nilai *forgetting factor* yang sama. Hal ini berlaku untuk proses pengurangan *noise* mesin diesel dan *noise* sepeda motor. Nilai MSE terkecil dari pengurangan *noise* mesin diesel diperoleh dari pengujian dengan panjang filter 16 dan *forgetting factor* 0,99 yaitu sebesar 0,0402 dan sebesar 0,0732 untuk pengurangan *noise* sepeda motor.

Grafik SNR menunjukkan bahwa nilai SNR yang terbesar, yakni 19,7436 dB, dihasilkan oleh filter adaptif dengan panjang 16 dan *forgetting factor* 0,95 untuk pengurangan *noise* mesin diesel. Sedangkan pengurangan *noise* sepeda motor menghasilkan nilai

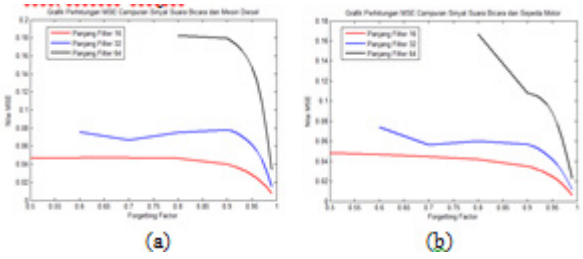
SNR terbesar (17,3054 dB) dengan menggunakan panjang filter 64 *forgetting factor* 0,98.

Grafik persentase *crest factor*, ΔC, menunjukkan bahwa nilai terkecil ΔC dihasilkan oleh filter adaptif dengan panjang filter 64 dan nilai *forgetting factor* 0,99 untuk pengurangan *noise* mesin diesel dan panjang filter 16 dan nilai *forgetting factor* 0,99 untuk pengurangan *noise* sepeda motor.

C. Pengujian dengan Sinyal Bicara

Pemfilteran *noise* dari sinyal bicara dilakukan secara simulasi pada perangkat lunak Matlab. Sinyal bicara murni dicampur dengan salah satu *noise* kemudian sinyal *noise* itu sendiri dijadikan sebagai *input noise* referensi. Hasil pemfilteran adalah sinyal estimasi yang diharapkan menyerupai dengan sinyal bicara murni. ANC diuji dengan mengubah panjang filter dan *forgetting factor*.

Sinyal estimasi dibandingkan dengan sinyal murni dan kemudian diukur seberapa besar perbedaan yang tampak dari kedua sinyal tersebut. Pengukuran dilakukan dengan menghitung MSE, SNR dan persentase *crest factor* dari kedua sinyal.



Gambar 5. Nilai-nilai MSE dari Pengurangan *Noise* pada Campuran Sinyal Bicara dengan (a) *Noise* Mesin Diesel dan (b) *Noise* Sepeda Motor

Hasil pengujian parameter MSE, SNR dan *crest factor* dapat dilihat pada Gambar 5. Kedua macam *noise* menghasilkan nilai MSE, SNR dan persentase *crest factor* yang berbeda meskipun dicampurkan pada sinyal yang sama. Setiap panjang filter mampu mengurangi *noise* yang tercampur pada sinyal bicara namun menghasilkan besar pengurangan yang berbeda. Gambar 5 juga menunjukkan bahwa nilai MSE yang dihasilkan dengan panjang filter 16 lebih kecil dibandingkan dengan nilai MSE yang dihasilkan oleh panjang filter 32 dan panjang filter 64. *Forgetting factor* 0,99 menghasilkan nilai MSE terkecil dibanding dengan nilai MSE yang dihasilkan dengan nilai *forgetting factor* lainnya pada setiap panjang filter yang digunakan pada filter adaptif. Kecilnya nilai MSE yang hampir mendekati nol ini menunjukkan bahwa sinyal estimasi yang dihasilkan filter adaptif ini mirip dengan sinyal bicara murni.

Grafik SNR dari pengujian sinyal bicara menunjukkan bahwa nilai terbesar SNR (26,8805 dB) dihasilkan oleh filter adaptif RLS dengan panjang filter 16 dan *forgetting factor* 0,91 untuk pengurangan *noise* mesin diesel, serta untuk pengurangan *noise* sepeda motor, dengan menggunakan panjang filter 16 dan *forgetting factor* 0,96 diperoleh nilai SNR terbesar 21,4276 dB.

Grafik persentase *crest factor* dari pengujian sinyal bicara menunjukkan bahwa nilai-nilai terkecil %ΔC dihasilkan oleh filter adaptif RLS dengan panjang filter 16 dan *forgetting factor* 0,99.

VI. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah bahwa pengurangan *noise* sepeda motor dan *noise* mesin diesel dengan algoritma RLS ini menghasilkan sinyal estimasi bicara yang mendekati sinyal bicara murni dengan *noise* yang telah berkurang. Pengurangan ini dapat dilihat pada parameter MSE yang dihasilkan. Nilai MSE yang dihasilkan adalah 0,7% untuk pengurangan *noise* mesin diesel dan 0,6% untuk pengurangan *noise* sepeda motor. Kedua angka ini didapat dengan memilih panjang filter 16 dan *forgetting factor* 0,99.

Referensi

[1] W. Hioki, *Telecommunications Second Edition*. New Jersey, USA: Prentice Hall Inc, pp. 399-409, 1995.

[2] S.A. Prasetyowati, A. Susanto, T. Sriwidodo dan J. E. Istiyanto, "Adaptive LMS Noise Cancellation of Wideband Vehicle's Noise Signals," in *Proceedings of* ICGC-RCICT, 2010.

[3] H. Candra dan E. Setyaningsih, "Pengurangan *Noise* pada Suara Percakapan dengan Filter Adaptif Menggunakan Algoritma RLS (Suatu Eksperimen)," Jurusan Teknik Elektro, Universitas Tarumanagara, Jakarta, Laporan Penelitian, 2005.

[4] R. Watson and O. Downey, *The Little Red Book of Acoustics: A Practical Guide*, United Kingdom: Blue Tree Acoustics, pp. 44-45, 2008.

[5] J. Gnitecki, Z. Moussavi, and H. Pasterkamp, "Recursive Least Square Adaptive Noise Cancellation Filtering for Heart Sound Reduction in Lung Sounds Recordings," in *Proceedings of the 25th Annual Int. Conf. of* IEEE Engineering in Medicine and Biology Society, vol. 3, pp. 2416 – 2419, 2003.

[6] M. H. Hayes, *Statistical Digital Signal Processing and Modelling,* New York, USA: John Wiley & Sons Inc, 1996.

[7] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach*, *2nd ed*., New Jersey, USA: Prentice Hall Inc, pp. 662 – 665, 2002.

[8] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing,* 4th ed. New Jersey, USA: Pearson Education, Inc, 1996.

[9] V. K. Ingle and J. G. Proakis, *Digital Signal Processing Using MATLAB*, *2nd ed.,* Canada: Thomson Learning, 2007.

[10] E. Setyaningsih dan Hugeng, "Digitalisasi dan Pengurangan Derau dari Rekaman Musik pada Kaset Audio dengan Menggunakan Filter Adaptif dengan Algoritma *Normalized Least Mean Square*," Jurusan Teknik Elektro, Jakarta, Universitas Tarumanagara, Laporan Penelitian, 2006.

ISSN 2355-3286

Proxy Selector Berbasis Link-State

Edwin Tunggawan[1], Hargyo Tri Nugroho I.[2]
[1]Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
vcc.edwint@gmail.com, [2] hargyo@umn.ac.id
[2]Program Studi Sistem Komputer, Universitas Multimedia Nusantara, Tangerang, Indonesia
hargyo@umn.ac.id

Diterima 2 Mei 2013
Disetujui 16 Mei 2013

Abstrak—Pemilihan *proxy server* yang digunakan untuk menjelajah web seringkali dilakukan secara manual oleh pengguna. Proses ini dapat diotomatisasi dengan menggunakan aplikasi *proxy selector*. Pada penelitian ini, *proxy selector* dikembangkan dengan mengadaptasi algoritma *link-state*. Algoritma disesuaikan untuk pemilihan proxy. Aplikasi juga diberikan kemampuan untuk melakukan prediksi dengan menggunakan perhitungan *moving average* untuk menghitung perkiraan performa *proxy server* yang diuji. Untuk mendapatkan hasil perhitungan yang lebih akurat, diilakukan percobaan untuk menentukan panjang periode (*n*) *moving average* yang paling baik dengan menggunakan perhitungan *mean squared error* terhadap data. Dari hasil perhitungan tersebut, didapatkan panjang periode yang memberikan hasil prediksi terbaik pada data percobaan.

Keywords—*proxy selector*, *link-state*, *moving average*, *mean squared error*

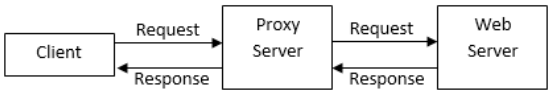
I. Pendahuluan

Pada saat ini, Internet telah digunakan secara luas untuk memenuhi berbagai kebutuhan Informasi masyarakat. Menjelajah web merupakan salah satu aktivitas yang paling banyak dilakukan oleh pengguna Internet. Pada saat mengakses suatu situs web, pengguna menggunakan sebuah *client host* yang akan mengakses sebuah *web server*. *Web server* kemudian akan memberikan respon berupa informasi yang diinginkan oleh pengguna. Respon akan dikirim ke *client*.

Ada kalanya dibutuhkan sebuah *proxy server* sebagai penengah di antara *client* dan *web server*. *Proxy server* memiliki beberapa manfaat bagi pengguna, antara lain untuk mempercepat akses ke laman web dan anonimitas.

Dalam sebuah instansi yang memiliki jaringan komputer cukup besar, *proxy server* dapat digunakan untuk mempercepat akses pengguna jaringan tersebut ke laman web dengan mengaktifkan fungsi *caching* pada *proxy server*. Dengan menyimpan *cache* dari laman web, *proxy server* dapat mengambil *cache* laman web yang diminta oleh *client* dan mengirimkannya ke *client* yang bersangkutan. *Request* tidak perlu selalu diteruskan ke *web server* karena *proxy server* menyimpan data dari laman web tersebut di dalam *cache* untuk diberikan ke *client* pada *request* selanjutnya. Dengan demikian, kecepatan akses ke laman web itu akan meningkat.

Terkadang pengguna perlu mengakses suatu laman web tanpa memberikan informasi mengenai lokasi dan *host* yang sedang digunakan ke *web server*. Pada situasi ini, pengguna dapat menggunakan *proxy server* sebagai perantara untuk mengakses laman web tersebut. *Request* dari *client* akan diteruskan oleh *proxy server* ke *web server*. Dalam situasi ini, yang akan tercatat sebagai *host* pengakses laman web di *web server* bukanlah *client*, melainkan *proxy server* yang menjadi perantara. Karena itu, *web server* tidak mendapatkan informasi mengenai *client* dan anonimitas pengguna tetap terjaga.



Gambar 1. *Proxy Server* Meneruskan *Request* dan *Response*

Untuk menggunakan *proxy server*, *client host* harus dikonfigurasi terlebih dahulu oleh pengguna agar mengakses laman web melalui *proxy server* yang dipilih. Dalam banyak kasus, terutama untuk mendapatkan anonimitas, pengguna harus memilih *proxy server* secara manual sebelum melakukan konfigurasi.

Penelitian ini mengadaptasikan algoritma *link-state* untuk melakukan pemilihan *proxy server*. Algoritma *link-state* umum digunakan oleh *router* untuk memilih rute terbaik dalam meneruskan data agar sampai ke *host* tujuan.

II. Algoritma Link-State

Salah satu algoritma *link-state routing* yang paling banyak digunakan adalah OSPF (*Open Shortest Path First*) [1]. Berikut ini adalah langkah-langkah yang dilakukan untuk menentukan rute pada OSPF [2].

- Setiap *router* akan bertukar informasi dengan

router-router tetangganya.

- Setelah mendapatkan informasi *link-state*, *router* akan menghitung jarak dengan menggunakan algoritma Dijkstra.
- *Router* akan mengirimkan paket data ke *router* lainnya untuk mengecek apakah kondisi *link-state* sesuai dengan informasi yang dimiliki.
- Jika terdapat perbedaan, *router* akan melakukan *update* pada *routing table* dan menyebarkannya ke *router* lainnya.

Moving Average dan Mean Squared Error

Moving average digunakan untuk melakukan analisa terhadap kecenderungan suatu nilai untuk bertambah, berkurang, atau tetap. *Moving average* umum digunakan dalam analisa pasar dan saham, dengan membandingkan nilai *moving average* data yang didapatkan dari perusahaan A dan perusahaan B. Data dihitung dengan interval (*n*) yang sama, kemudian hasilnya dianalisa untuk menentukan prediksi [3]. Berkut ini adalah rumus *simple moving average*.

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

Gambar 2. *Simple Moving Average*

Interval dari *moving average* berpengaruh terhadap seberapa baik hasil prediksi yang diberikan. Untuk mengukur seberapa baik hasil prediksi dari suatu perhitungan *moving average*, dapat digunakan perhitungan *mean squared error* (MSE). Berikut ini adalah rumus MSE.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2.$$

Gambar 3. *Mean Squared Error*

Prediksi yang paling baik akan menghasilkan nilai MSE yang paling rendah [4].

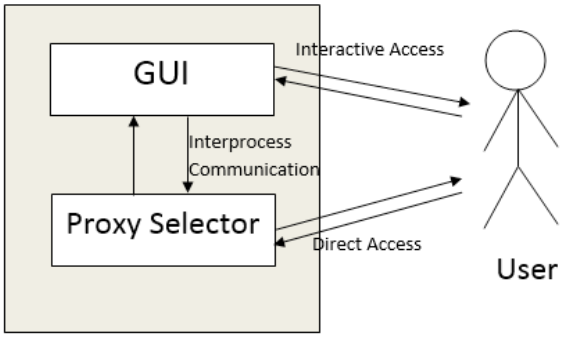
Pemilihan Server dengan Pendekatan Probabilistik

Dalam melakukan pemilihan *server* untuk keperluan *load-balancing*, pendekatan probabilistik dapat digunakan untuk menghitung kemungkinan-kemungkinan pemilihan *server* oleh *client* [5]. Perhitungan tersebut bermaksud untuk mengantisipasi agar tidak terlalu banyak *client* yang mengakses satu *server* secara bersamaan sehingga *server* tersebut mengalami *overload*.

Pendekatan probabilistik juga dapat digunakan dalam jaringan komunikasi P2P, agar *client* dapat secara otomatis memilih konten yang diperkirakan akan dibutuhkan untuk proses selanjutnya dari *peer* yang telah dipilih berdasarkan perkiraan kinerja dari *peer* yang bersangkutan. Perkiraan tersebut dibuat berdasarkan kinerjanya pada beberapa operasi terakhir [6].

III. Struktur Aplikasi

Aplikasi dibagi menjadi dua bagian, yaitu *proxy selector* yang dituliskan dengan menggunakan bahasa pemrograman Python dan GUI yang dituliskan dengan menggunakan bahasa pemrograman C#. Pengguna dapat mengakses langsung program *proxy selector* yang berbasis *console* atau menggunakan GUI yang melakukan *interprocess communication* dengan *proxy selector*.



Gambar 4. Interaksi antara GUI, *Proxy Selector*, dan Pengguna

Proxy server dapat melakukan *caching* untuk halaman-halaman yang telah diakses, dan jika *client* me-*request* halaman tersebut maka *proxy server* akan memberikan data yang tersimpan di dalam *cache*. Jika *client* me-*request* halaman yang belum terdapat pada *cache*, maka *proxy server* akan meneruskan *request* ke *web server* untuk meminta halaman yang diinginkan, kemudian memberikannya pada *client*. Karena itu, pengujian untuk memilih *proxy server* dibagi menjadi dua modus: tanpa *cache* dan dengan *cache*.

Topologi *logic* pada sistem komunikasi antara *client*, *web server*, dan *proxy server* lebih sederhana jika dibandingkan dengan topologi sistem *routing*. Karena itu, algoritma *link-state* yang digunakan dalam aplikasi ini adalah bentuk yang telah disederhanakan dan diadaptasikan untuk sistem komunikasi *client*, *proxy server*, dan *web server*.

- *Proxy selector* mengecek daftar *proxy server* dari sebuah *text file*.
- Setelah memuat daftar *proxy server*, *proxy selector* akan mengecek *history* dari kondisi *link-state* dan prediksi yang telah didapatkan sebelumnya.
- *Proxy selector* melakukan pengujian performa kepada *proxy server* yang terdaftar, mengukur kecepatannya, dan menghitung prediksi performanya untuk beberapa saat ke depan dengan *moving average* dan mencatat hasilnya ke dalam *text file*.
- Program memberikan *output* berupa data *testing history* dan prediksi untuk dibaca oleh pengguna.

Pada poin ketiga dari empat poin di atas, ditambahkan perhitungan *moving average* untuk melakukan prediksi performa *proxy server* untuk beberapa waktu ke depan. Berbeda dengan sistem *routing* yang memiliki jumlah *node* lebih banyak untuk melakukan lompatan (*hop*), jumlah *node* pada topologi *logic* untuk sistem komunikasi *client*, *proxy server*, dan *web server* hanya memiliki satu *node* untuk melakukan lompatan. Tetapi kecepatan *node* tersebut cenderung tidak stabil jika dibandingkan dengan *node* pada sistem *routing* jika yang digunakan adalah *proxy server* yang alamatnya disebarkan di Internet. Karena itulah ditambahkan kemampuan untuk melakukan prediksi pada aplikasi ini.

IV. Percobaan dan Analisa Data

Untuk mendapatkan data, dilakukan pengujian terhadap tiga buah *proxy server* berikut.

- 202.137.22.182:8080 (*Proxy 1*)
- 116.213.51.82:80 (*Proxy 2*)
- 118.98.35.251:8080 (*Proxy 3*)

Pengumpulan data dibagi menjadi dua bagian, yaitu pengumpulan data dengan memperhitungkan *cache* dan pengumpulan data tanpa memperhitungkan *cache*. Untuk kedua bagian pengumpulan data tersebut dilakukan empat kali percobaan terhadap *proxy server*. Pada masing-masing percobaan, *proxy selector* akan mengirimkan *request* sebanyak 60 kali dengan *timeout* 10 detik kepada ketiga *proxy server*.

Setelah data didapatkan, dilakukan perhitungan dengan menggunakan *moving average* dengan interval (*n*) dalam *range* 2-10 untuk setiap percobaan yang dilakukan. Berdasarkan hasil perhitungan *moving average* itu, dibuat perhitungan MSE untuk mencari panjang interval yang memberikan prediksi paling baik untuk data hasil percobaan.

Tanpa Cache

Berikut ini adalah nilai *n* yang memberikan nilai MSE minimum pada setiap percobaan pada pengumpulan data tanpa memperhitungkan *cache* dari *proxy server*.

Tabel 1. Interval dengan MSE Minimum

Percobaan	*Proxy 1*	*Proxy 2*	*Proxy 3*
I	10	10	10
II	10	10	10
III	10	10	10
IV	10	7	7

Pada tabel terlihat jika pada percobaan I sampai dengan percobaan III, prediksi terbaik pada semua *proxy server* didapatkan dengan menggunakan nilai *n* sebesar 10 pada perhitungan *moving average*. Pada percobaan IV, nilai 10 memberikan prediksi terbaik pada *Proxy 1* sementara untuk *Proxy 2* dan *Proxy 3* prediksi terbaik didapatkan dengan nilai *n* sebesar 7.

Dengan demikian, dari 12 kasus pada percobaan I sampai dengan percobaan IV, nilai *n* sebesar 10 memberikan prediksi terbaik pada 10 kasus, sementara nilai *n* sebesar 7 hanya memberikan prediksi terbaik pada 2 kasus. Maka nilai 10 dianggap sebagai panjang interval yang paling baik.

Dengan Cache

Berikut ini adalah nilai *n* yang memberikan nilai MSE minimum pada setiap percobaan pada pengumpulan data dengan memperhitungkan *cache* dari *proxy server*.

Tabel 2. Interval dengan MSE Minimum

Percobaan	*Proxy 1*	*Proxy 2*	*Proxy 3*
I	10	10	10
II	10	10	10
III	10	10	10
IV	10	7	7

Nilai *n* yang menghasilkan nilai MSE minimum dengan memperhitungkan *cache* ini lebih bervariasi dibandingkan perhitungan tanpa *cache*. Pada percobaan I, *Proxy 1*, *Proxy 2*, dan *Proxy 3* masing-masing memiliki hasil prediksi terbaik pada nilai *n* sebesar 7, 8, dan 9. Pada percobaan II, hasil prediksi terbaik untuk ketiga *proxy server* tersebut didapatkan pada nilai *n* sebesar 10, 7, dan 10. Untuk percobaan III, prediksi terbaiknya didapatkan pada nilai *n* sebesar 10, 9, dan 10, sementara untuk percobaan IV nilai *n* yang memberikan prediksi terbaiknya adalah sebesar 7, 10, dan 10.

Dengan demikian, dari 12 kasus pada percobaan I sampai dengan percobaan IV, nilai *n* sebesar 10 memberikan prediksi terbaik pada 6 kasus, sementara nilai *n* sebesar 9 pada 2 kasus, nilai *n* sebesar 8 pada 1 kasus, dan nilai *n* sebesar 7 memberikan prediksi terbaik pada 2 kasus. Karena lebih banyak memberikan hasil prediksi yang terbaik, nilai 10 masih dianggap sebagai panjang interval yang paling baik untuk perhitungan dengan *cache*.

V. Uji Performa Aplikasi

Pengujian performa aplikasi dilakukan dengan menggunakan AMD CodeAnalyst pada komputer dengan *processor* Intel Core2 Duo T6600 dan RAM sebesar 2 GB. Penggunaan rata-rata RAM untuk aplikasi secara keseluruhan menurut AMD CodeAnalyst adalah 0,97% dari total RAM yang tersedia, yaitu sebesar 19,4 MB. Aplikasi GUI-nya sendiri berukuran 372 KB, dan terbagi menjadi dua bagian utama: *splash screen* dan *form* utama aplikasi.

Pada saat AMD CodeAnalyst digunakan hanya untuk menguji *splash screen*, penggunaan RAM dari aplikasi adalah 0% dari total RAM yang tersedia. Penggunaan RAM sebesar 0 B untuk menjalankan aplikasi adalah tidak mungkin, karena aplikasi tersebut harus dimuat di RAM. Jika dilakukan perhitungan, persentase ukuran aplikasi dari total RAM adalah (372.000 / 2.000.000) x 100% = 0,000186%. Nilai tersebut cukup dekat dengan nilai 0% yang diberikan

oleh AMD CodeAnalyst, sehingga dapat disimpulkan jika penggunaan RAM oleh aplikasi sejauh ini adalah normal.

Saat AMD CodeAnalyst digunakan untuk menguji *form* utama aplikasi dan menjalankan *proxy selector* dengan GUI selama kurang lebih 5 menit, AMD CodeAnalyst memberikan hasil penggunaan RAM sebesar 0,98% dari total RAM yang tersedia, yaitu sebesar 19,6 MB. Nilai ini jauh lebih besar dibandingkan konsumsi sumber daya RAM oleh *splash screen* karena pada *form* utama terdapat lebih banyak fungsi, variabel, *class*, dan *object* yang digunakan untuk operasi dan lebih banyak *dynamic-linking library* yang ikut dimuat ke RAM.

Jika *proxy selector* dijalankan tanpa GUI, AMD CodeAnalyst memberikan hasil penggunaan RAM sebesar 0% dari total RAM yang tersedia. *Proxy selector* tanpa GUI ini adalah sebuah *script* Python yang berukuran 7 KB yang dijalankan dengan *interpreter* Python yang berukuran 26 KB.

VI. Kesimpulan

Algoritma *link-state* telah diadaptasikan untuk aplikasi *proxy selector*, dengan menggunakan *moving average* untuk membantu pemilihan *proxy server* dengan pendekatan probabilistik. Berdasarkan perhitungan MSE dari data hasil eksperimen, panjang interval (*n*) *moving average* yang memberikan prediksi terbaik adalah 10. Prediksi yang terbaik dapat diberikan oleh panjang interval yang berbeda jika perhitungan dilakukan dengan menggunakan kumpulan data lain.

Pada sebagian besar kasus dalam eksperimen yang dilakukan, nilai *n* sebesar 10 paling banyak memberikan prediksi yang terbaik menurut perhitungan MSE. Nilai tersebut adalah nilai tertinggi dalam *range* nilai *n* yang diuji (2 sampai dengan 10). Karena itu, besar kemungkinan terdapat panjang interval *moving average* yang memberikan hasil prediksi lebih baik dari $n = 10$ untuk eksperimen ini jika *range* nilai *n* yang diuji diperbesar sampai lebih dari 10.

Referensi

[1] A. Shaikh, D. Wang, G. Li, J. Yates, and C. Kalmanek, *An Efficient Algorithm for OSPF Subnet Aggregation*, 1998.

[2] Nortel Networks., *NetKnowledge: Routing*, Nortel Networks, 2002.

[3] N. E. Hwa, *Different Uses of Moving Average (MA)*, 2007.

[4] V. K. Boken, *Forecasting Spring Wheat Yield Using Time Series Analysis: A Case Study of Canadian Prairie*, Agronomy Journal, 2000.

[5] N. Bartolini, G. Bongiovanni, and S. Silvestri, *Distributed Server Selection and Admission Control in Replicated Web Systems*, IEEE Computer Society, 2007.

[6] L. D'Acunto, N. Chiluka, T. Vinkó, and H. Sips, *BitTorrent-like P2P approaches for VoD: A comparative study, Elsevier, 2013.*