Majalah Ilmiah Teknologi Elektro, Vol. 19, No. 1, Januari - Juni 2020
DOI: https://doi.org/10.24843/MITE.2020.v19i01.P08

Rancang Bangun Aplikasi *File Transfer* Menggunakan *Library* ZeroMQ

I Gusti Ngurah Yoga Pawitra[1], Rukmi Sari Hartati[2], Yoga Divayana[3]

Submission: 11-02-2020, Accepted: 08-06-2020

***Abstract*— Sending data with large data sizes between two or more computers in a company is affected by various factors, both the network connection of the two computers to the data exchange method used, one method that can be used is the ZeroMQ library. The method of sending files using the ZeroMQ method is relatively faster because the architecture is brokerless or without middleware as an intermediary so that the process of sending files can be done quickly and can blow up the resources used. The ZeroMQ method was originally created to send messages, but we can add file conversion algorithms to binary numbers, so the messages sent are binary numbers from encoded files. This application is built on the Python programming language so that it is easier to develop again. The results of the design of file transfer applications using the ZeroMQ library performed on two server computers and client computers explain that this method can be used to send files and from the results of the test the time required and the latency value is relatively faster and more stable.**

***Intisari*— Pengiriman data dengan ukuran data yang besar antar dua komputer atau lebih pada sebuah perusahaan dipengaruhi oleh berbagai faktor, baik koneksi jaringan dari kedua komputer hingga metode pertukaran data yang digunakan, salah satu metode yang dapat digunakan adalah dengan *library* ZeroMQ. Metode pengiriman *file* menggunakan metode ZeroMQ tergolong lebih cepat karena pada arsitekturnya yang *brokerless* atau tanpa *middleware* sebagai perantaranya sehingga proses pengiriman *file* dapat dilakukan dengan cepat dan dapat menghembat sumber daya yang digunakan. Metode ZeroMQ awalnya dibuat untuk mengirimkan pesan, tetapi kita dapat menambahkan algoritma konversi *file* menjadi bilangan biner, sehingga pesan yang dikirim merupakan bilangan *biner* dari *file* yang telah di *encode*. Aplikasi ini dibangun pada Bahasa pemrograman Python sehingga lebih mudah untuk dikembangkan kembali. Hasil dari rancang bangun aplikasi *file* transfer menggunakan *library* ZeroMQ yang dilakukan pada dua komputer server dan komputer client menjelaskan bahwa metode ini dapat digunakan untuk mengirim *file* dan dari hasil uji coba waktu yang diperlukan dan nilai *latency* nya relatif lebih cepat dan stabil**

***Kata Kunci*— *File* Transfer, ZeroMQ, *Brokerless*, Python.**

[1] *Mahasiswa, Magister Teknik Elektro Universitas Udayana Gedung Pascasarjana Universitas Udayana, Jl. PB Sudirman Denpasar-Bali 80232 (tlp: 0361-555225; fax: 0361-4321982; e-mail: yoga.pawitra@student.unud.ac.id*

[2, 3] *Dosen, Magister Teknik Elektro Universitas Udayana Gedung Pascasarjana Universitas Udayana, Jl. PB Sudirman Denpasar-Bali 80232 (tlp: 0361-555225; fax: 0361-4321982; e-mail: rukmisari@unud.ac.id), yoga@unud.ac.id)*

I. PENDAHULUAN

Teknologi informasi memiliki banyak manfaat untuk memudahkan pekerjaan manusia, khususnya pada perusahaan modern yang tentunya memanfaatkan teknologi informasi untuk mempermudahkan bisni proses pada perusahaan tersebut. Perusahaan modern digital menjalankan sebagian besar pekerjaannya menggunakan komputer dan pada perusahaan besar yang memiliki banyak cabang tentunya memiliki sistem terintegrasi untuk dapat saling bertukar data antar kantor, data yang dikirimpun beragam mulai dari *text*, gambar, video hingga data cadangan antara *server* dan *client* [1]. Pertukaran *file* yang sangat besar di perusahaan - perusahaan adalah hal yang umum terutama di perusahaan keuangan [2]. Semakin berkembangnya teknologi akan berdampak pada pertumbuhan data yang berlipat ganda dari waktu ke waktu [3]. Sistem pertukaran data antar komputer membutuhkan koneksi internet atau LAN, pada kasus pertukaran data dengan kapasitas yang besar tentunya memerlukan metode yang tepat untuk dapat mengatasinya. Aplikasi transfer *file* dianggap sebagai salah satu sumber data terbesar pada jaringan [4]. *Condor*, yang umum digunakan dalam *grid Computational* yang dapat dimanfaatkan untuk mentransfer *file*, menyediakan mekanisme untuk berbagi sumber daya komputasi ratusan ribu komputer, memiliki kekurangan dalam mentransfer data [5].

ZeroMQ dapat menjadi solusi untuk dapat melakukan pertukaran data antar komputer dengan kapasitas yang besar dengan cepat. Zero MQ merupakan salah satu metode yang dapat digunakan pada file transfer *protocol*, Zero MQ dapat melakukan pertukaran data dengan cepat karena memiliki arsitektur yang sederhana sehingga proses yang dilakukan menjadi lebih cepat, selain itu Zero MQ dapat menyesuaikan *core* yang digunakan, sehingga pada komputer yang memiliki kemampuan *multi core* dapat melakukan pertukaran data lebih cepat akan tetapi metode ini juga tidak memerlukan *memory* yang banyak, hanya membutuhkan sepasang *pages* pada *memory* yang ada, sehingga dapat berjalan pada platform dengan *memory* terbatas. Penggunaan sedikit *memory* juga penting untuk mengoptimalkan penggunaan L1i *cache*. Ketika ukuran kode cukup kecil, *processor* dapat menahan seluruh kode messaging dalam L1i cache menghindari akses lambat pada *physical memory* untuk mendapatkan potongan kode baru. ZeroMQ dapat digunakan untuk menghubungkan kode dalam bahasa apapun seperti C, C ++, JAVA, NET, Python dan pada platform apapun seperti Linux, Windows dan MAC OS [6].

II. LITERATUR REVIEW

Berikut ini akan dijelaskan mengenai teori pendukung yang menjadi landasan penelitian ini, diantaranya:

A. File Transfer

Ilmu *File Transfer Protocol* (FTP) adalah protokol jaringan standar yang digunakan untuk mentransfer *file* komputer dari satu host ke host lain melalui jaringan berbasis TCP, seperti Internet. FTP dibangun di atas arsitektur *server* klien dan menggunakan kontrol terpisah dan koneksi data antara klien dan *server*. Pengguna APP dapat mengotentikasi dirinya dengan menggunakan protokol masuk yang jelas, biasanya dalam bentuk nama pengguna dan kata sandi, namun dapat terhubung secara anonim jika *server* dikonfigurasi untuk mengizinkannya Untuk transmisi aman yang melindungi *username* dan *password*, dan mengenkripsi isinya, FTP sering diamankan dengan SSL / TLS (FTPS) [7]

B. ZeroMQ

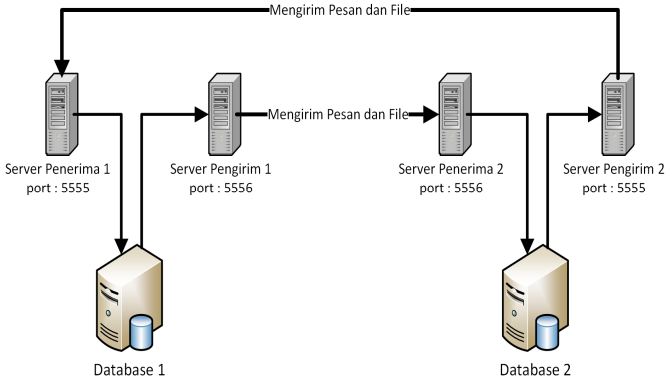
ZeroMQ merupakan implementasi dari *lightweight* messaging dengan *socket* seperti API. ZeroMQ sangat cepat, mampu mengirim/menerima pesan hingga 4.100.000 pesan per detik. Hanya membutuhkan sepasang *pages* pada *memory* yang ada, sehingga dapat berjalan pada platform dengan *memory* terbatas. Penggunaan sedikit *memory* juga penting untuk mengoptimalkan penggunaan L1i cache. Ketika ukuran kode cukup kecil, *processor* dapat menahan seluruh kode *messaging* dalam L1i *cache* menghindari akses lambat pada *physical memory* untuk mendapatkan potongan kode baru. Sebagai lapisan *transport* yang sederhana dan mudah digunakan, ZeroMQ sangat menyederhanakan pemrograman *Socket* dengan *mode* komunikasi yang fleksibel dan beragam [8]. Arsitektur ZeroMQ juga telah mendukung konsep dasar penggunaan RESTful API [9].

C. Python

Alternatif gratis untuk platform komersial adalah bahasa pemrograman Python, yang tersedia di bawah lisensi *open source*. Python adalah bahasa *scripting* dengan sintaks yang lurus ke depan dan mudah dipelajari. Perpustakaan ilmiah seperti numpy dan scipy diimplementasikan secara internal di C, sehingga analisis matematika dan rutinitas manipulasi data dilakukan secara efisien. Python telah mendapatkan popularitas yang signifikan untuk proyek *open source* terbuka, terutama dalam aplikasi ilmiah. Karena berbagai macam perpustakaan tersedia secara bebas dalam Python, aplikasi Python dapat dengan mudah diperluas dengan perpustakaan pihak ketiga. Mereka juga dapat diparalelkan pada kelompok komputasi tanpa lisensi atau kendala kompatibilitas. Akibatnya, banyak alat yang baru-baru ini dikembangkan untuk analisis sistem tenaga diterapkan dengan Python [10].

D. Gambaran Umum Aplikasi

Aplikasi ini berfungsi untuk mengirimkan *file* antara dua komputer atau dalam kasus yang lebih besar dapat menangani lebih dari dua komputer. *File* dikirim secara langsung tanpa bantuan *broker* atau *middleware*, proses pengiriman *file* dilakukan dengan *library* ZeroMQ yang dapat mengirimkan pesan dengan cepat. Gambaran umum dari aplikasi ini dapat dilihat pada gambar 1.



Gambar 1: Gambaran Umum Aplikasi

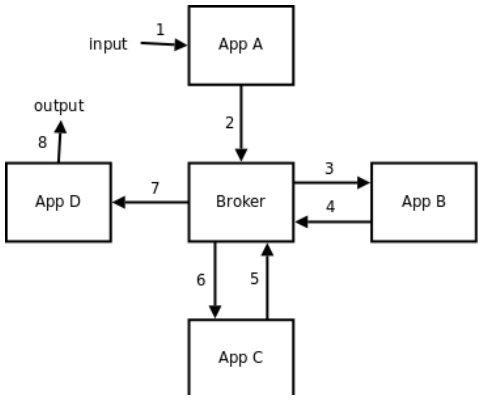
Proses dimulai dari penerima pertama menerima pesan yang berisikan isi pesan dan *file*, pesan akan disimpan pada *database*, ZeroMQ akan bekerja setelah terjadi penambahan oleh penerima pertama, apabila pesan yang diterima penerima pertama sama dengan pesan yang telah ditentukan pada tb_service maka dengan otomatis menambahkan data pesan balasan dan *file* yang telah ditentukan, tetapi bila pesan tidak sesuai maka pesan berhenti pada tb_inbox saja. Penerima pertama akan menangkap data baru yang belum dikirim dengan waktu jeda yang telah ditentukan dan dikirim ke penerima 2, program akan jalan terus menerus dan akan saling mengirim pesan atau *file* apabila terjadi penambahan data, pada penelitian ini akan hanya akan melakukan pertukaran data antara dua komputer.

III. METODELOGI PENELITIAN

Perancangan arsitektur, *flowchart* pada perancangan aplikasi *file* transfer menggunakan *library* ZeroMQ dan metode-metode yang akan digunakan pada aplikasi ini akan dijelaskan pada bab ini.

A. Broker

Sebagaian besar arsitektur sistem pengiriman pesan dibedakan oleh *server* pesan (*broker*) di tengah atau dapat disebut arsitektur (*star*) dan (*hub*). Setiap aplikasi terhubung ke *broker* pusat sehingga tidak ada aplikasi yang berbicara langsung ke aplikasi lain dan semua komunikasi dilewatkan melalui *broker*. Contoh arsitektur *broker* dapat dilihat pada gambar berikut 2.

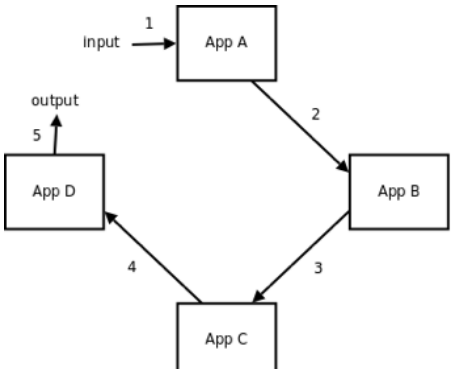


Gambar 2: Arsitektur *Broker*

Kelemahan model *broker* yaitu membutuhkan jumlah komunikasi jaringan yang berlebihan dan semua pesan harus diteruskan melalui *broker* yang dapat menyebabkan *broker* berubah menjadi hambatan dari keseluruhan sistem. Kotak *broker* dapat terbebani hingga 100% sementara kotak lainnya kurang dimanfaatkan, bahkan hampir tidak digunakan sepanjang waktu.

B. No Broker

Aplikasi ini dapat saling mengirim pesan tanpa menggunakan *broker* atau *middleware* sehingga rute yang dilalalukan untuk mengirim sebuat pesan menjadi lebih singkat, arsitektur dari metode *no broker* dapat dilihat pada gambar 3.



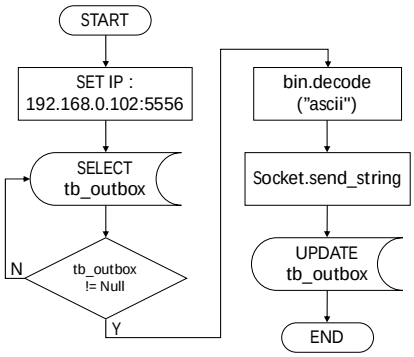
Gambar 3: Arsitektur No Border

Arsitektur tanpa *broker* atau *brokerless* memungkinkan aplikasi untuk mengirim pesan lebih cepat karena pesan yang dikirim tanpa melalui perantara *middleware* dan dengan pesan yang berjalan searah tanpa adanya *broker* dapat menghindari terjadinya *bottleneck*. Metode ini sangat sederhana hanya menggunakan alur data satu arah dan dapat menghemat sumber daya yang digunakan.

C. Flowchart

Flowchart merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program [11]. Penjelasan mendetail mengenai setiap proses yang dilakukan akan dijelaskan dengan *flowchart*, terdapat 2 proses yang dijelaskan secara mendetail, yaitu proses komputer *server* pesan dan komputer *client*.

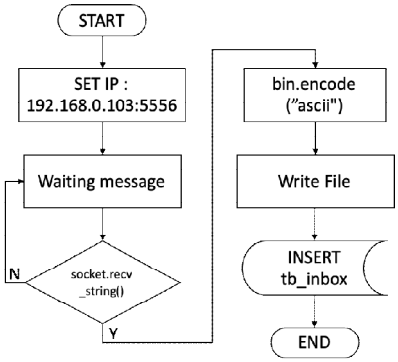
1) Komputer Server



Gambar 4: *Flowchart Server Client*

Proses awal dari *server* pengirim adalah dengan mengatur IP *address* dan *port* yang akan digunakan, pada percobaan ini IP yang digunakan yaitu 192.168.0.102 dengan port 5556. Selanjutnya mengambil data tb_outbox pada *database*, apabila tb_outbox kosong maka proses akan kembali ke pada cek tb_outbox, sedangkan apabila terdapat pesan yang belum dieksekusi maka proses selanjutnya adalah mengambil *file* dan merubahnya ke format ASCII, kemudian pesan dan *file* yang telah berupa *biner* akan dikirim dengan metode PUSH kepada *client*. Setelah proses pengiriman berhasil maka tb_outbox akan diupdate dan menandakan bahwa proses pengiriman telah berhasil dilakukan.

2) Komputer Client



Gambar 5: Flowchart Komputer *Client*

Proses pada komputer *client* hampir sama dengan proses pada *server*, proses diawali dengan mengatur IP *address* 192.168.0.3 dan *port* 5556. Selanjutnya aplikasi akan menunggu pesan *yang* akan dikirim oleh *server* pengirimin. Apabila terdapat pesan yang dikirim oleh *server* pengirim maka aplikasi pada komputer *client* akan menangkap pesan yang dikirim. Selanjutnya pesan akan di *encode* dari format ASII sehingga komputer *client* dapat mengenali *file* yang dikirim dan setelah itu aplikasi menulis *file* pada komputer *client* dan memasukan data ke tb_outbox.

IV. HASIL DAN PEMBAHASAN

A. Tampilan Interface

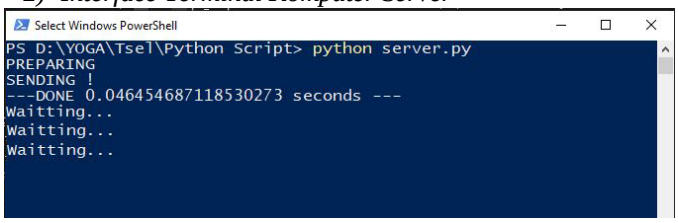
Penelitian ini menghasilkan tiga tampilan *window* yang berbeda pada setiap prosesnya, tiga tampilan *window* terdiri dari satu tampilan aplikasi *desktop* dan dua tampilan aplikasi yang dijalankan pada Terminal.

1) Interface Form Send Message

Gambar 6: *Form Send Message*

Form Send Message merupakan sebuah *form* yang digunakan untuk memilih *server* yang akan menerima pesan, memasukan pesan dan memilih *file* yang akan dikirim. *Form Send Message* dikembangkan dengan *library* Python Tkinter.

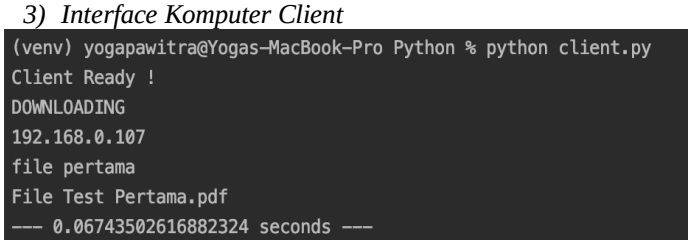
2) Interface Terminal Komputer Server



Gambar 7: Terminal Komputer *Server*

Komputer *server* menjalankan aplikasi pada terminal, sehingga proses pengiriman pesan dan *file* akan ditampilkan pada terminal. Status dari proses pengiriman pesan dan *file* akan ditampilkan pada terminal ini hingga proses pengiriman telah berhasil dilakukan. *Server* pengirim pada percobaan ini menggunaakn OS Windows sehingga akan menampilkan Terminal Windows.

3) Interface Komputer Client



Gambar 8: Terminal Komputer *Client*

Komputer *client* secara langsung akan menerima pesan dan *file* yang telah dikirim oleh komputer *server*. Aplikasi *client* dijalankan pada Terminal dan menampilkan informasi proses pengiriman pesan dan *file*. Percobaan menerima pesan dan *file* dilakukan pada OS macOS.

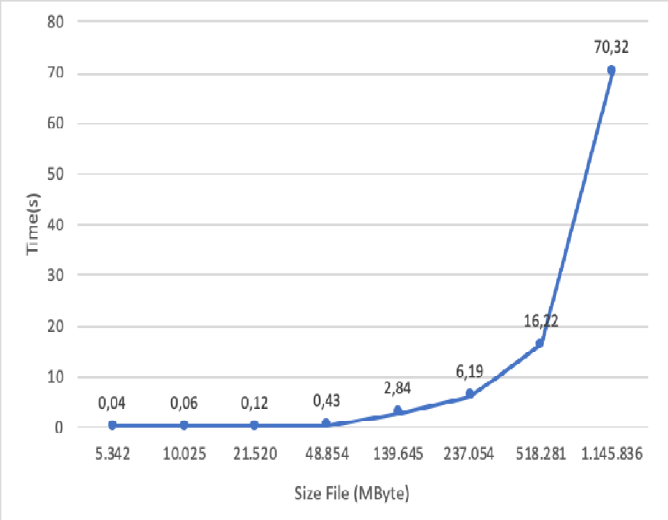
B. Pengujian Komputer Server

Pengujian mengirimkan *file* pada komputer *server* dilakukan dengan mengirimkan *file* dengan ukuran yang berbeda-beda. Hasil pengukuran pengujian dari komputer *Server* dapat dilihat pada tabel 1.

TABEL I
PENGUKURAN WAKTU PENGUJIAN KOMPUTER *SERVER*

No	Time (s)	Size (MB)	Ping (ms)
1	0,04	5.342	3
2	0,06	10.025	10
3	0,12	21.520	17
4	0,43	48.854	4
5	2,84	139.645	2
6	6,19	237.054	2
7	16,22	518.281	50
8	70,32	1.145.836	2

Hasil pengujian yang diukur yaitu durasi waktu yang diperlukan selama proses pengiriman *file* (*Time*) dengan satuan detik, ukuran *file* yang dikirim (*Size*) dengan satuan MegaByte dan Ping yang diukur merupakan antara komputer *server* dan *client* yang terhubung menggunakan jaringan *wifi* sehingga hasil ping yang diperoleh kurang stabil yang dihitung dalam satuan *millisecond*.



Gambar 9. Grafik Pengukuran Waktu Pengujian Komputer Server

Grafik pengukuran waktu pengujian pada komputer *server* menjelaskan bahwa proses pengiriman *file* dengan menggunakan ZeroMQ relatif cepat dan stabil. Percobaan dilakukan mulai pada ukuran *file* 5 MByte hingga 1.145 MByte.

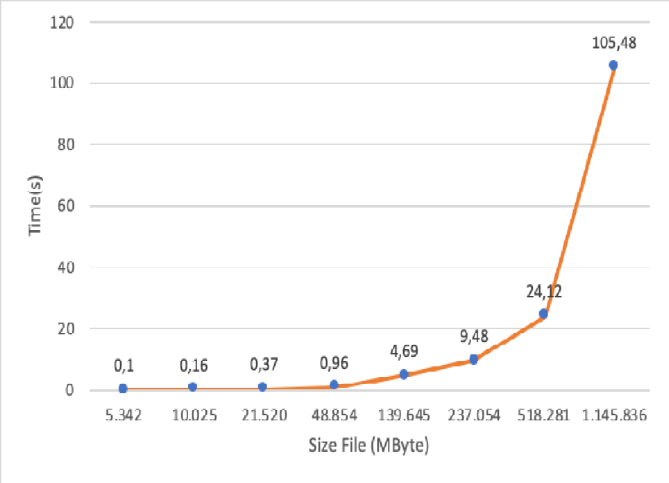
C. Pengujian Komputer Client

Pengujian pada komputer *client* dilakukan sebanyak *file* dikirimkan oleh komputer *server*. Hasil dari pengujian komputer *client* dapat dilihat pada tabel 2.

TABEL II
PENGUKURAN WAKTU PENGUJIAN KOMPUTER *CLIENT*

No	Time (s)	Size (MB)	Ping (ms)
1	0,1	5.342	3
2	0,16	10.025	10
3	0,37	21.520	17
4	0,96	48.854	4
5	4,69	139.645	2
6	9,48	237.054	2
7	24,12	518.281	50
8	105,48	1.145.836	2

Pengujian pada komputer *client* dilakukan dengan ukuran *file* dan ping yang sama akan tetapi menghasilkan waktu yang berbeda. Waktu yang dicatat adalah waktu sejak *file* mulai dikirim hingga *file* terkirim. Hasil pengujian pada komputer client juga menunjukan waktu yang cukup cepat dan stabil.



Gambar 10. Grafik Pengukuran Waktu Pengujian Komputer *Client*

Hasil pengujian yang dilakukan pada komputer *client* menjelaskan bahwa durasi waktu yang diperlukan untuk proses menerima *file* relatif lebih lama dibandingkan dengan mengirim *file*, hal ini disebabkan karena pada saat menerima file, komputer *client* membutuhkan waktu lebih lama untuk men-*download file* yang dikirim dan menyimpannya pada harddisk, akan tetapi hal ini berbanding terbalik pada *file* dengan berukuran besar.

D. Menghitung Latency

Untuk menghitung *latency* pengiriman *file* antara komputer dapat menggunakan rumus berikut

$$Ln = In - On$$

Ln = *Latency* dari n-th pesan (dalam detik)
In = Waktu ketika n-th pesan diterima
On = Waktu ketika n-th pesan dikirim

Hasil perhitungan *latency* pengiriman *file* yang telah dilakukan dapat dilihat pada tabel 3.

TABEL IIII
PENGUKURAN *LATENCY* PENGIRIMAN

No	On (s)	In (s)	Latency (s)
1	0,04	0,1	0,06
2	0,06	0,16	0,1
3	0,12	0,37	0,25
4	0,43	0,96	0,53
5	2,84	4,69	1,85
6	6,19	9,48	3,29
7	16,22	24,12	7,9
8	70,32	105,48	35,16

Hasil pengukuran *latency* yang dilakukan dengan mengurangi waktu ketika pesan diterima dengan waktu ketika pesan dikirim, hasil dari pengukuran menunjukan nilai yang cendung kecil.

V. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil perancangan hingga pengujian pada aplikasi *file* transfer menggunakan *library* ZeroMQ telah berhasil dibuat dengan baik, hasil dari pengujian aplikasi menunjukan kemampuan aplikasi yang dapat mengirim *file* dengan cukup stabil, dari delapan kali pengujian yang dilakukan telah dapat diselesaikan dengan baik dengan rata-rata waktu dibutuhkan untuk mentransfer *file* adalah 18,17 detik dengan *latency* rata-rata 6,14. Metode ZeroMQ dapat mengirim *file* dengan ukuran yang tidak terbatas, akan tetapi metode ini sangat bergantung pada sumber daya *memory* yang digunakan, sehingga untuk dapat mengirim *file* dengan ukuran yang besar perlu dilakukannya *split file* agar tidak membebani *memory* pada satu waktu. Adapun kekurangan dari aplikasi ini dan dapat menjadi pengembangan untuk penelitian selanjutnya adalah mencoba melakukan pengiriman *file* dengan lebih dari satu komputer, karena mengirim *file* dengan *library* ZeroMQ dapat dilakukan ke banyak komputer pada ip *address* yang berbeda-beda.

REFRENSI

[1] I. P. A. E. D. Udayana and N. P. Sastra, "Perbandingan Performansi Pengamanan File Backup LPSE Menggunakan Algoritma DES Dan AES," *Majalah Ilmiah Teknologi Elektro*, vol. 15, no. 1, pp. 111-117, 2016.

[2] W. Sardjono and T. Pujadi, "Performance evaluation of systems Managed File Transfer in banking industry using IT Balanced Scorecard," in *International Conference on Information Management and Technology (ICIMTech)*, Bandung, Indonesia, 2016.

[3] I. P. A. P. Wibawa, I. D. Giriantari and M. Sudarma, "Komputasi Paralel Menggunakan Model Message Passing Pada SIM RS (Sistem Informasi Manajemen Rumah Sakit)," *Majalah Ilmiah Teknologi Elektro*, vol. 17, no. 3, pp. 439-444, 2017.

[4] Y. S. Aljarbou, "Large File Transfer Using Multicast and P2P for Error Checking and Correction," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, Riyadh, Saudi Arabia, Saudi Arabia, 2019.

[5] S. C. U. o. T. G. C. Gao Ying Sch. of Comput. Sci. & Eng., L. GuanYao and H. JianCong, "A New Method of File Transfer in Computational Grid Using P2P Technique," in *International Conference on Network Computing and Information Security*, Guilin, China, 2011.

[6] I. G. J. Arissaputra, I. M. Sukarsa, P. W. Buana and N. W. Wisswani, "Binary Log Design for One-Way Data Replication with ZeroMQ," *I.J. Modern Education and Computer Science*, vol. 10, pp. 22-30, 2018.

[7] D. Ruwaida and D. Kurnia, "RANCANG BANGUN FILE TRANSFER PROTOCOL (FTP) DENGAN PENGAMANAN OPEN SSL PADA JARINGAN VPN MIKROTIK DI SMKS DWIWARNA. CESS," *Journal of Computer Engineering, System and Science*, vol. 3, no. 1, pp. 45-49, 2019.

[8] L. Zhang, X. Wang and Y. Yaoqi, "Time Delay Performance Analysis of Distributed Communication Platform Based on ZeroMQ," in *International Conference on Communications, Information System and Computer Engineering (CISCE)*, Haikou, China, China, 2019.

[9] J. Moore, A. Arcia-Moret, P. Yadav, R. Mortier, A. Brown and D. McAuley, "Zest: REST over ZeroMQ," in *IEEE International Conference on Pervasive Computing and Communications Workshops*

(PerCom Workshops), Kyoto, Japan, Japan, 2019.

[10] L. Thurne, "Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510 - 6521, 2018.

[11] I. M. Sukarsa, "APLIKASI KONVERSI FLOWCHART KE KODE PROGRAM BAHASA PEMROGRAMAN PL/SQL MYSQL," *Majalah Ilmiah Teknologi Elektro*, vol. 8, no. 2, pp. 2503-2372, 2019.