Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 6, Juni 2018, hlm. 2235-2242 http://j-ptiik.ub.ac.id

Analisis Perbandingan Kinerja Protokol Websocket dengan Protokol SSE pada Teknologi *Push Notification*

Panser Brigade Muhammad[1], Widhi Yahya[2], Achmad Basuki[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]panser.brig@gmail.com, [2]widhi.yahya@ub.ac.id, [3]abazh@ub.ac.id

Abstrak

Teknologi *push notification* memungkinkan pengguna (*client*) mendapatkan informasi secara berkala. Informasi yang dikirimkan oleh *push notification* berupa notifikasi seperti pada jejaring sosial atau situs berita. Terdapat banyak teknologi yang dapat diterapkan pada pengiriman notifikasi. Salah satunya adalah pengiriman notifikasi menggunakan protokol Websocket dan *Server-Sent Events* (SSE). Protokol Websocket menerapkan komunikasi *full-duplex* sehingga baik *server* atau *client* dapat mengirim dan menerima data secara bersamaan. Sedangkan protokol SSE menerapkan komunikasi *half-duplex* sehingga *client* hanya dapat menerima data yang dikirim *server* secara berkala. Penelitian ini dilakukan untuk membandingkan kinerja protokol Websocket dengan SSE terkait metode pengiriman kedua protokol pada teknologi *push notification*. Kedua protokol akan diimplementasikan dan dilakukan pengujian berdasarkan skenario yang telah ditentukan. Hasil yang diperoleh adalah rata-rata *delay* dan penggunaan CPU pada protokol SSE lebih kecil dibandingkan protokol Websocket.

Kata kunci: *Push Notification*, Websocket, Server-Sent Events, *Smartphone*, Android

Abstract

Push notification technology allows users (clients) to get information periodically. Information submitted by push notification in the form of notifications such as on social network or news sites. There are many technologies that can be applied to notification delivery. One of them is notification delivery using Websocket protocol and Server-Sent Events (SSE). The Websocket protocol implements full-duplex communication so that either the server or client can send and receive data simultaneously. While the SSE protocol implements half-duplex communication so that the client can only receive data sent server periodically. This study was conducted to compare the performance of Websocket protocol with SSE related to method of delivery of both protocols on push notification technology. Both protocols will be implemented and tested based on predetermined scenarios. The results obtained are the average delay and CPU usage in SSE protocol is smaller than Websocket protocol.

***Keywords**: Push Notification*, Websocket, Server-Sent Events, *Smartphone*, Android

1. PENDAHULUAN

Seiring dengan berkembangnya teknologi, derasnya arus pertukaran informasi sebanding dengan meningkatnya kebutuhan manusia akan informasi tersebut. Hal ini didukung oleh munculnya perangkat canggih yaitu komputer dan smartphone. Perangkat tersebut memungkinkan manusia sebagai pengguna untuk memperoleh akses dari informasi yang tersebar secara gratis atau cuma-cuma. Dari banyaknya informasi yang tersebar, terdapat beberapa informasi yang harus didapatkan secara cepat dan kontinyu seperti kebijakan politik terbaru hingga tagihan rekening masing-masing pengguna. Untuk mendapatkan informasi tersebut secara berkala dan kontinyu dibutuhkan fitur notifikasi yang sudah tersedia baik pada perangkat smartphone ataupun komputer. Fitur notifikasi ini biasa disebut dengan teknologi *push notification*.

Push notification adalah teknologi berbasis standar teknis yang berfungsi untuk mengurangi kelebihan informasi dengan mengirimkan informasi user melalui internet secara teratur (Guo & Liu, 2013). Mekanisme dari teknologi ini adalah *server* mengirim notifikasi kepada *client* tanpa harus diminta terlebih dahulu. Mekanisme pengiriman *push notification* diatur oleh komponen yang biasa disebut *broker*

(Brüstel & Preuss, 2012). Perangkat android menggunakan layanan Google Cloud Messaging (GCM) sebagai *broker*. Dengan adanya teknologi *push notification* para pengguna perangkat *smartphone* ataupun komputer dimudahkan dengan dikirimnya informasi secara berkala sehingga tidak perlu membuka aplikasi secara terus-menerus untuk mendapatkan informasi terbaru.

Terdapat banyak teknologi yang dapat diterapkan pada teknologi *push notification*. Salah satunya adalah pengiriman notifikasi dengan menggunakan Websocket dan *Server-Sent Event* (SSE). Websocket adalah protokol yang menerapkan komunikasi *full-duplex* antara *server* dan *client* (Zhang & Shen, 2013). Protokol websocket memiliki dua arah komunikasi yang berbeda yaitu untuk pengiriman dari server ke client dan pengiriman dari client ke server. *Server-Sent Event* adalah protokol yang menerapkan komunikasi satu arah dari *server* ke *client*. Mekanisme kerja SSE adalah dengan membuka satu koneksi dalam waktu yang cukup lama untuk mengirimkan notifikasi secara terkini. Protokol SSE lebih mudah diterapkan dan tidak memerlukan konfigurasi lebih untuk mendapatkan komunikasi satu arah yang lebih stabil (Estep, 2013).

Penelitian sebelumnya yang berjudul "Mobile HTML5: Efficiency and Performance of Websockets and Server-Sent Events" oleh Elliot Estep (Estep, 2013). Salah satu aspek yang ingin dicapai oleh Elliot Estep adalah untuk mengetahui performa protokol yang diuji pada sisi *client*. Parameter yang digunakan adalah penggunaan *resource* pada sisi *client*. Sedangkan penelitian yang akan dilakukan oleh penulis adalah untuk mengetahui penggunaan *resource* kinerja *push service* pada sisi server. Sehingga diperoleh hasil protokol mana yang mempunyai penggunaan CPU lebih kecil. Penelitian kedua yang menjadi acuan penulis adalah "Internet Based Communication using Server Push" yang dilakukan oleh Pragya (Pragya, 2016). Penelitian yang dilakukan oleh Pragya bertujuan untuk mengetahui mekanisme mana yang lebih baik untuk diterapkan pada pengiriman notifikasi dengan menggunakan long-polling, websocket atau SSE berdasarkan dari teknologi pengiriman masing-masing protokol. Sedangkan pada penelitian yang akan dilakukan oleh penulis adalah untuk membandingkan kinerja protokol websocket dan SSE berdasarkan parameter *delay* dan penggunaan CPU.

Berdasarkan studi kasus diatas, penelitian yang akan dilakukan adalah dengan melakukan analisis perbandingan kinerja protokol websocket dengan SSE pada teknologi *push notification*. Penelitian akan dilakukan dengan menggunakan *framework* flask, *library* Socket.IO, bahasa pemrograman python dan modul psutil pada ruang lingkup jaringan lokal. Untuk *server* menggunakan laptop dengan sistem operasi Ubuntu dan untuk *client*menggunakan *smartphone* dengan sistem operasi android. Pengujian akan dilakukan menggunakan parameter *delay* dan penggunaan CPU. Hasil yang diperoleh akan dianalisis untuk mengetahui kinerja dari protokol websocket dan protokol SSE terkait metode pengiriman berdasarkan parameter uji.

2. KAJIAN KEPUSTAKAAN

Pada penelitian sebelumnya yang berjudul "Pengembangan Push Notification Menggunakan Websocket*"* oleh Putra menerapkan protokol Websocket untuk pengiriman notifikasi. Ruang lingkup yang digunakan adalah jaringan lokal. Perangkat yang digunakan adalah smartphone dengan sistem operasi dan laptop dengan sistem operasi windows dan linux. Pengujian yang dilakukan pada penelitian ini bertujuan untuk mendapatkan kinerja pengiriman Websocket dengan menggunakan parameter delay. Kesimpulan dari penelitian ini adalah terdapat delay yang signifikan antara perangkat smartphone dan laptop pada pengiriman notifikasi dengan menggunakan protokol Websocket (Putra, 2016).

Pada penelitian sebelumnya yang berjudul "Mobile HTML5: Efficiency and Performance of Websockets and Server-Sent Events" oleh Estep menerapkan protokol Websocket dan SSE. Dari kedua protokol yang diteliti, penulis mengacu pada penelitian protokol SSE yang diterapkan pada browser perangkat smartphone. Tujuan dari penelitian ini untuk mengetahui performa EventSource dan memperhitungkan dampak dari performa EventSource pada beberapa perangkat smartphone dengan menggunakan beberapa browser yang berbeda. Kesimpulan dari penelitian ini adalah kinerja yang tidak seragam dan sangat bervariasi tergantung pada browser dan konfigurasi pada jaringan (Estep, 2013).

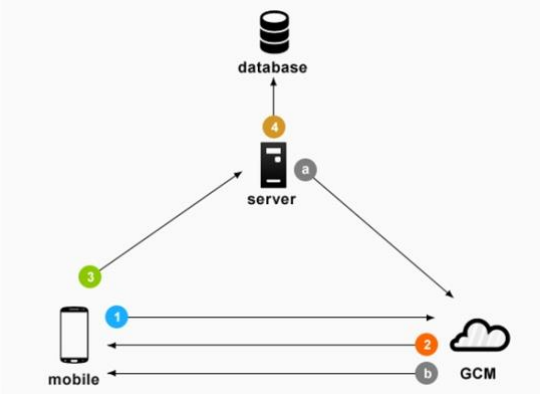
Pada penelitian sebelumnya yang berjudul

"Internet Based Communication using Server Push" oleh Pragya. Penelitian ini menggunakan tiga teknologi *push* yaitu long-polling, server-sent event dan Websocket. Dari tiga teknologi yang digunakan, penulis mengacu pada penelitian protokol SSE dan Websocket sebagai pembanding dari penelitian yang akan dilakukan. Tujuan dari penelitian ini adalah untuk membandingkan kinerja pengiriman notifikasi dari tiga teknologi *push* yang sudah disebut sebelumnya. Hasil dari penelitian yang dilakukan menyebutkan bahwa pengiriman yang dilakukan oleh SSE dapat juga dilakukan oleh Websocket dengan performa yang berbeda. Pada beberapa aspek terdapat perbedaan antara protokol SSE dan Websocket seperti tingkat kesulitan penerapan, *delay* dan *server-loading* . Kesimpulan yang didapat adalah protokol Websocket lebih baik dari long-polling dan SSE karena protokol Websocket dapat melakukan komunikasi dua arah dan memiliki *delay* yang lebih kecil dibandingkan long-polling dan SSE (Pragya, 2016).

3. PUSH NOTIFICATION

Push notification adalah teknologi berbasis standar teknis yang berfungsi untuk mengurangi kelebihan informasi dengan mengirimkan informasi user melalui internet secara teratur (Guo & Liu, 2013). Pada perangkat *smartphone*, *push notification* adalah pesan dari *server* sebagai penyedia informasi yang muncul pada perangkat yang mana *client* tidak perlu berada pada aplikasi yang mengirimkan notifikasi atau sedang dalam menggunakan perangkat tersebut (Urbanairship.com, 2017).

Tiap smartphone memiliki sistem operasi yang berbeda dan masing – masing sistem operasi memiliki teknologi *push notification* yang berbeda. Pada penelitian ini sistem operasi pada smartphone yang digunakan adalah operasi sistem berbasis android karena sistem operasi Android lebih sederhana dan memiliki *hardware* yang dapat bersaing. Teknologi *push notification* pada *smartphone* yang berbasis android diatur oleh layanan Google Cloud Messaging (GCM) (Brüstel, 2012). GCM berfungsi sebagai *broker*. Untuk mendapatkan layanan *push notification* dari GCM, *client* harus terlebih dahulu terdaftar ke *server* untuk mendapatkan tanda pengenal (Ding, J, et al., 2014). Tanda pengenal (*token*) diperlukan agar notifikasi yang dikirim oleh *server* tepat terkirim pada *client* yang mempunyai tanda pengenal (token) tersebut.



Gambar 1 Mekanisme kerja*push notification* menggunakan layanan GCM
Sumber: **(Webgeometrics.com)**

Gambar 1 menjelaskan teknologi *push notification* dengan menggunakan layanan GCM sebagai *broker* pada smartphone berbasis android. Sistem kerja dari *push notification* adalah sebagai berikut:

- *Client* mendaftar pada GCM sebagai *broker*
- Setelah terdaftar *broker* memberikan tanda pengenal (*token*) kepada *client*.*Token* berfungsi agar notifikasi yang dikirim tepat kepada pemilik *token*
- Setelah *token* didapatkan dari *broker*, *client* akan mengirimkan token ke *server* aplikasi
- *Token* yang didapat dari *client* akan disimpan ke *database*

Mekanisme pengiriman notifikasi dari *server* ke *client* adalah:

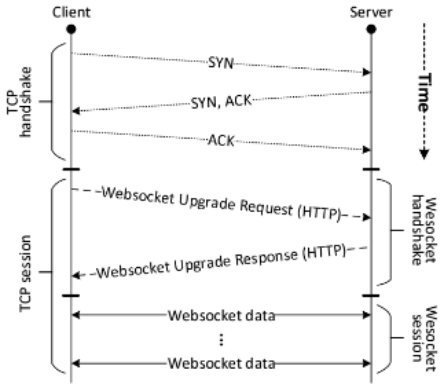
- *Server* mengirimkan notifikasi kepada *broker*
- *Broker* menerima notifikasi dari *server* dan dikirimkan ke *client* sesuai dengan token yang disimpan di database *server* aplikasi.

4. WEBSOCKET

Websocket adalah protokol yang dikembangkan oleh HTML5 yang menerapkan komunikasi *full – duplex* antara *server* dengan *client* (Zhang & Shen, 2013). Protokol Websocket dirancang untuk diterapkan di *web browser* dan *webserver*, tetapi dapat digunakan oleh aplikasi *client* atau *server*. Protokol Websocket sendiri berbasis pada *Transfer Control Protocol* (TCP).

Pada dasarnya Websocket dirancang untuk menyerupai TCP kecuali pada segi fungsionalitas. Hal ini memungkinkan *server*

Websocket untuk berbagi port dengan *server* HTTP, sehingga membuat Websocket mudah disebarkan dalam beberapa skala (Estep, 2013).

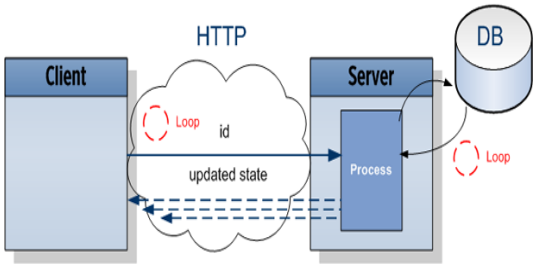


Gambar 2 Mekanisme kerja protokol Websocket
Sumber : **(Skvorc, Horvat, & Srbljic, 2014)**

Gambar 2 menunjukan proses *handshake* antara *client* dengan *server* pada protokol Websocket. Pertama, *client* akan meminta koneksi Websocket pada *server*. Kemudian *server* akan merespon permintaan koneksi dari *client*. Setelah proses *handshake* terbentuk protokol Websocket dapat mengirim dan menerima data dalam model dual channel atau dalam jalur berbeda pada waktu yang sama (Zhang & Shen, 2013). *Server* akan secara aktif mengirim data berupa notifikasi kepada *client*. Proses pengiriman notifikasi akan terus berjalan hingga diberikan perintah untuk memutuskan koneksi pada sisi *server*. Pada penelitian ini, *push server* Websocket sekaligus menjadi *broker* sehingga pengiriman langsung dari *server* kepada *client*. Ketika koneksi *client* dengan *server* terputus, terdapat pemberitahuan pada sisi *server*.

5. SERVER-SENT EVENTS (SSE)

Protokol server-sent event adalah protokol yang juga dikembangkan oleh HTML5. Berbeda dengan Websocket yang menerapkan komunikasi dua arah atau *full – duplex*, protokol SSE lebih memfokuskan pada komunikasi satu arah atau *half – duplex* dari server ke client. Protokol SSE lebih menyerupai HTTP *Streaming*, yang mana membuka satu koneksi selama mungkin untuk pengiriman notifikasi secara akurat atau *real time*. SSE lebih mudah diterapkan dan tidak memerlukan konfigurasi lebih untuk mendapatkan komunikasi satu arah yang lebih stabil (Estep, 2013).



Gambar 3 Mekanisme kerja protokol SSE
Sumber: **(Real-time Web Apps, 2017)**

Gambar 3 menunjukan mekanisme kerja protokol SSE antara *client* dengan *server*. Awalnya *client* melakukan permintaan koneksi dan membentuk proses yang merujuk kondisi terbaru pada database. Saat koneksi telah terbentuk, *server* akan mengirimkan informasi dari waktu ke waktu. Pada sisi *client* akan mendapatkan pembaruan informasi dari waktu ke waktu selama koneksi masih terbentuk (Pragya, 2016). SSE memiliki fitur *automatic reconnection*. Jadi apabila koneksi *client* atau *browser* terputus maka akan langsung mencoba untuk terhubung kembali secara otomatis ke server. Tetapi pada sisi *server* tidak terdapat pemberitahuan apabila *client* terputus dari *server*.

6. PERANCANGAN DAN IMPLEMENTASI

Simulasi yang akan dirancang oleh penulis adalah simulasi mekanisme kerja pengiriman notifikasi yang dikirimkan oleh *server* ke *client* dengan merujuk pada tugas akhir Andrias Yudianto Putra yang berjudul "Pengembangan *Push notification* Menggunakan Websocket". Pada penelitian ini, penulis akan menambahkan perancangan dengan menggunakan protokol SSE untuk dapat membandingkan hasil antara kedua protokol sesuai dengan tujuan penelitian ini.

Penelitian ini dibagi menjadi 2 yaitu implementasi protokol Websocket dan protokol SSE kedalam mekanisme pengiriman pada *push notification* pada jaringan lokal yang sama (WLAN).

Tabel 1 Tabel Kebutuhan *Server* dan *Client*

	Server	***Client***
	Laptop	***Smartphone***
Bahasa Pemrograman	Python	Javascript
Framework	Flask	-
Library	Socket.IO	-

Sistem Operasi	Ubuntu 16.04	Android

Pada tabel 1 dapat dilihat bahwa perangkat yang berperan sebagai *server* adalah laptop dengan menggunakan sistem operasi Linux Ubuntu 16.04. Implementasi pengiriman notifikasi menggunakan Websocket dan SSE pada sisi *server* menggunakan bahasa pemrograman Python. Selain itu juga menggunakan *framework* Flask dan *library* Socket.IO. *Framework* Flask berfungsi untuk merancang *server* yang didukung oleh *library* Socket.IO.

Untuk perangkat yang berperan sebagai *client* adalah *smartphone* dengan menggunakan sistem operasi Android. Bahasa pemrograman yang digunakan adalah Javascript. Javascript berfungsi untuk merancang halaman situs yang akan diakses oleh *client* melalui *browser*.

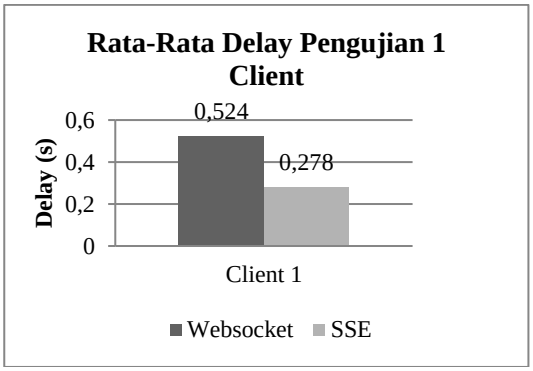
7. HASIL DAN PEMBAHASAN

Pengujian ini dilakukan dengan cara mengirim notifikasi pada 1 *client*, 3 *client* dan 6 *client*. Hasil yang diperoleh dari 3 skenario tersebut akan dianalisis untuk mendapatkan rata-rata *delay* pada tiap skenario. *Delay* diperoleh dengan menggunakan *script* pencatat waktu dan operasi perhitungan rata-rata pada sisi *client* sehingga tidak perlu melakukan penghitungan nilai rata-rata delay tiap skenario saat pengolahan data.

Pengujian *monitoring resource* dilakukan dengan cara melakukan instalasi modul psutil untuk memperoleh data penggunaan CPU yang berupa *log* pada *server*. Hasil penggunaan CPU diperoleh dengan cara mengurangi nilai CPU setelah dilakukan pengiriman notifikasi dengan nilai CPU sebelum dilakukan pengiriman. Nilai CPU yang ditampilkan pada sisi *server* dalam bentuk persen.

7.1. Pengujian Pengiriman Pada 1 *Client*

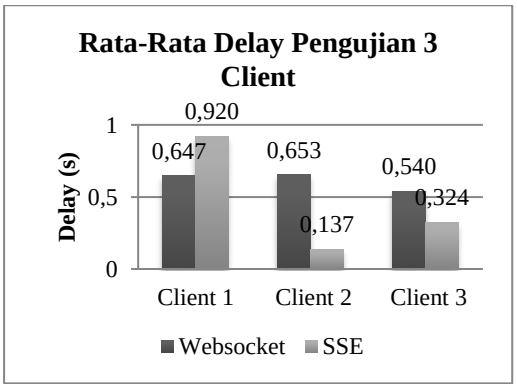
Gambar 4 adalah grafik rata-rata *delay* pada pengujian menggunakan 1 *client* uji. Berdasarkan grafik pada gambar 5.1 dapat dilihat bahwa terjadi perbedaan delay antara pengiriman menggunakan Websocket dengan SSE. Selisih delay antara Websocket dengan SSEadalah 0.245s.



Gambar 4 Grafik rata-rata *delay* pengiriman pada 1 *client*

7.2. Pengujian Pengiriman Pada 3 *Client*

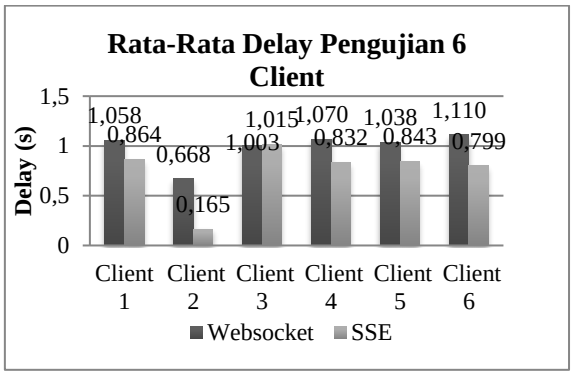
Pengujian selanjutnya adalah pengujian pengiriman notifikasi dengan menggunakan 3 *client* uji.Gambar 5 adalah grafik rata-rata *delay* pengiriman notifikasi dengan menggunakan 3 *client* uji.



Gambar 5 Grafik rata-rata *delay* pengiriman pada 3 *client*

Berdasarkan grafik pada gambar 5 dapat dilihat bahwa terdapat perbedaan *delay* dari *client* 1 dengan merujuk pada pengujian menggunakan 1 *client*. *Client* 1 memiliki *delay* sebesar 0.523s pada pengujian 1 client untuk protokol Websocket dan untuk SSE memiliki nilai *delay* sebesar 0.278s. Sedangkan pada pengujian dengan menggunakan 3 *client*, pengiriman menggunakan Websocket memiliki *delay* sebesar 0.646s dan untuk pengiriman menggunakan SSE memiliki *delay* sebesar 0.919s. Pada grafik diketahui bahwa *delay* terkecil terdapat pada *client* 2 dengan pengiriman menggunakan SSE dan *delay* terbesar terdapat pada *client* 1 dengan pengiriman menggunakan SSE.

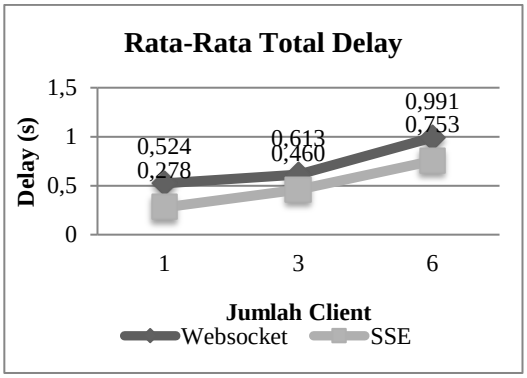
7.3. Pengujian Pengiriman Pada 6 *Client*



Gambar 6 Grafik rata-rata *delay* pengiriman pada 6 *client*

Pengujian selanjutnya adalah pengujian pengiriman notifikasi dengan menggunakan 6 *client* uji. Hasil dari pengujian ditunjukkan pada gambar 6. Pada gambar 6 dapat dilihat perbedaan delay yang sangat besar dibandingkan dengan pengujian dengan menggunakan 1 *client* dan 3 *client*. Pada pengiriman menggunakan Websocket nilai rata-rata delaynya mencapai 1s untuk semua *client*. Sedangkan untuk pengiriman menggunakan SSE, nilai delay tertinggi hanya pada *client* 3 yaitu sebesar 1.014s. Untuk nilai delay tertinggi dari keseluruhan pengujian pengiriman notifikasi terdapat pada *client* 6 dengan menggunakan Websocket. Nilai delay yang didapat sebesar 1.110s.

7.4. Analisis Pengujian Pengiriman Notifikasi



Gambar 7 Grafik rata-rata total *delay* pengiriman notifikasi

Dari grafik pada gambar 7 dapat diketahui bahwa pertambahan jumlah *client* mempengaruhi pertambahan nilai rata-rata *delay*.

Pada pengujian dengan menggunakan 1 *client*, *delay* pengiriman menggunakan protokol Websocket 0.523s. Ketika pengujian dengan menggunakan 3 *client* nilai rata-rata *delay* yang didapat sebesar 0.613s. Terjadi penambahan *delay* sebesar 0.089s. Pada pengiriman dengan menggunakan 6 *client*, rata-rata *delay* yang didapat adalah sebesar 0.991s. Jadi total kenaikan rata-rata *delay* sebesar 0.467s.

Sedangkan untuk pengiriman notifikasi menggunakan SSE diperoleh*delay* yang lebih kecil dibandingkan pengiriman notifikasi menggunakan Websocket untuk seluruh *client*. Pada pengujian pengiriman notifikasi dengan menggunakan 1 *client* uji, didapatkan rata-rata *delay* sebesar 0.278s. Terdapat kenaikan rata-rata *delay* sebesar 0.181s pada pengujian pengiriman dengan menggunakan 3 *client* uji. Terjadi kenaikan rata-rata *delay* lagi pada pengiriman dengan menggunakan 6 *client* uji sebesar 0.292s. Jadi total pertambahan *delay* yang diperoleh saat pengujian menggunakan SSE adalah sebesar 0.474s.

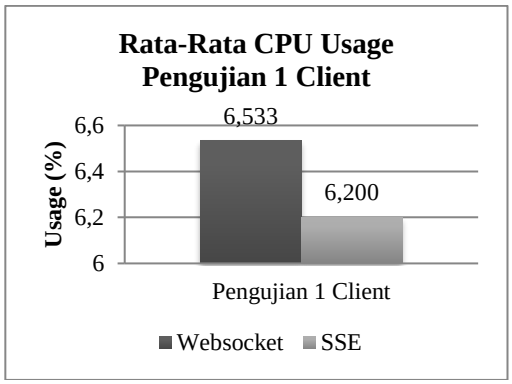
Perbedaan rata-rata *delay* ini disebabkan olehmetode pengiriman masing-masing protokol.Rata-rata *delay* pada protokol SSE lebih kecil dibandingkan dengan protokol Websocket karena metode pengiriman oleh protokol SSE lebih sederhana dibandingkan Websocket. Pengiriman notifikasi pada protokol SSE bekerja dengan langsung mengirimkan notifikasi ketika terdapat perintah untuk mengirim hingga terdapat perintah berhenti dari *server*. Ketika notifikasi diterima *client* langsung ditampilkan oleh pada halaman antar-muka.

Sedangkan untuk protokol Websocket ketika *server* diberikan perintah untuk mengirim, notifikasitidak langsung dikirim seperti pada protokol SSE. Protokol Websocket mengumpulkan dan membungkus beberapa notifikasi terlebih dahulu, setelah itu kumpulan notifikasi baru dikirim ke *client*. Ketika notifikasi diterima oleh *client*, kumpulan notifikasi tersebut akan dibongkar dan disusun ulang. Setelah itu baru ditampilkan pada halaman antar-muka. Protokol Websocket membutuhkan waktu yang lebih banyak untuk menampilkan *delay* pada *client* dibandingkan dengan protokol SSE dikarenakan proses penyusunan, pembongkaran dan penyusunan ulang notifikasi yang dikirim.

Selain perbedaan metode pengiriman dari protokol Webscoket dan SSE, penambahan jumlah *client* pada tiap skenario juga memengaruhi *delay* yang diperoleh.*Delay* yang diperoleh dari pengujian pengiriman notifikasi

mengalami kenaikan ketika ditambahkan jumlah *client* yang diuji.

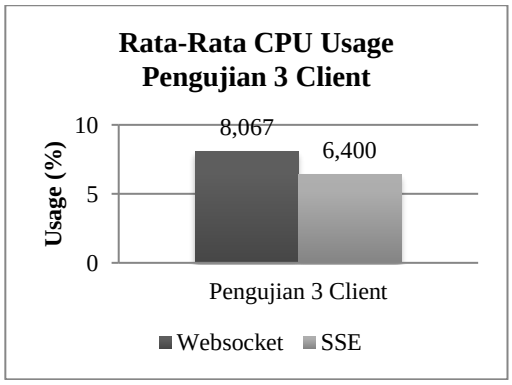
7.5. Pengujian Monitoring Resource Pada 1 *Client*



Gambar 8 Grafik rata-rata penggunaan CPU pada 1 *client*

Pada grafik pada gambar 8 dapat dilihat bahwa perbedaan penggunaan CPU antara pengiriman notifikasi menggunakan Websocket dengan SSE adalah sebesar 0.33%.

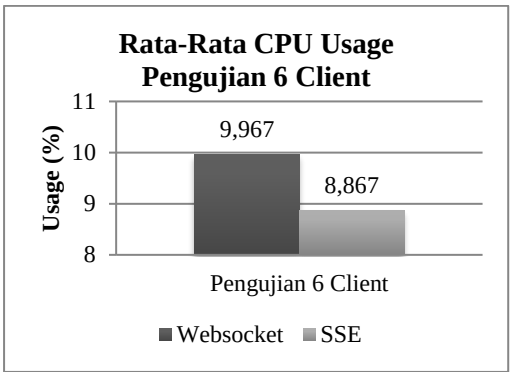
7.6. Pengujian Monitoring Resource Pada 3 *Client*



Gambar 9 Grafik rata-rata penggunaan CPU pada 3 client

Pada grafik pada gambar 9 dapat dilihat bahwa terjadi kenaikan penggunaan CPU yang cukup besar saat pengiriman notifikasi oleh protokol Websocket. Pada pengiriman dengan menggunakan 1 *client* nilai penggunaan CPU pada Websocket bernilai 6.533%. Sedangkan pada pengiriman menggunakan 3 *client* nilai penggunaan CPU Websocket bernilai 8.066%. Terjadi kenaikan penggunaan CPU sebesar 2.736%. Untuk penggunaan CPU pada pengiriman notifikasi pada server SSE dengan menggunakan 3 *client* uji mengalami kenaikan sebesar 0.2%. Nilai penggunaan CPU pada *server* SSE jauh lebih kecil dibandingkan dengan penggunaan CPU pada *server* Websocket dengan selisih 2.536%.

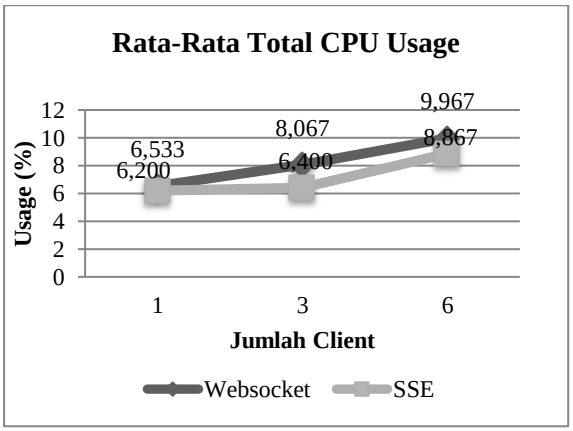
7.7. Pengujian Monitoring Resource Pada 6 *Client*



Gambar 10 Grafik rata-rata penggunaan CPU pada 6 client

Berdasarkan grafik pada gambar 10 *server* Websocket memiliki nilai penggunaan CPU yang paling tinggi dengan nilai penggunaan CPU mencapai 9.966%. Sedangkan *server* SSE hanya mencapai 8.866%.

7.8. Analisis Pengujian Monitoring Resource



Gambar 11 Grafik rata-rata total delay pengiriman notifikasi

Berdasarkan grafik pada gambar 11 dapat dilihat bahwa nilai penggunaan CPU pada kedua protokol yang diujikan meningkat seiring dengan penambahan jumlah *client* yang diuji.

Perbedaan kenaikan penggunaan CPU pada gambar 11 terjadi karena perbedaan metode pengiriman protokol Websocket dan protokol SSE. Jika pada *server* SSE, *resource* yang digunakan tidak sebanyak Websocket karena *server* SSE langsung mengirim notifikasi kepada *client*. Jadi *resource* yang dipakai oleh *server* hanya untukmengirim notifikasi.

Sedangkan pada *server* Websocket

membutuhkan *resource* yang lebih besa rkarena saat melakukan pengiriman, *server* Websocket terlebih dahulu mengumpulkan notifikasi. Setelah melakukan pengumpulan maka notifikasi dikirim kepada *client*. Karena proses pengumpulan notifikasi dan membagi pengirimannya menjadi beberapa kali, nilai penggunaan CPU pada *server* menjadi lebih tinggi saat pengujian pengiriman notifikasi pada 6 *client*.

8. KESIMPULAN

Dari hasil pengujian yang didapatkan terdapat perbedaan delay yang tidak terlalu besar antara pengiriman notifikasi menggunakan protokol Websocket dengan protokol SSE. Hal ini disebabkan oleh metode pengiriman masing-masing protokol. Rata-rata *delay* pada protokol SSE lebih kecil dibandingkan dengan protokol Websocket karena metode pengiriman oleh protokol SSE lebih sederhana. Rata-rata total *delay* keseluruhan *client* uji pada protokol SSEadalah sebesar 0.497s sedangkan rata-rata total *delay* keseluruhan *client* uji pada protokol Websocket adalah sebesar 0.709s. Penambahan jumlah *client* uji juga memengaruhi *delay* yang diperoleh. Semakin banyak client yang dikirim notifikasi, maka *delay* akan semakin besar.

Untuk perbedaan nilai penggunaan CPU diperoleh hasil yang cukup besar dikarenakan perbedaan metode pengiriman kedua *server*. *Server* SSE membutuhkan *resource* yang lebih sedikit yaitu sebesar 7.155% daripada *server* Websocket dengan nilai penggunaan CPU sebesar 8.188%. Hal ini dikarenakan metode pengiriman *server* SSE yang lebih sederhana. Selain itu juga terdapat pengaruh antara jumlah *client* uji dengan nilai penggunaan CPU. Semakin banyak jumlah *client*, maka nilai penggunaan CPU akan semakin besar.

9. DAFTAR PUSTAKA

Brüstel, J., & Preuss, T. (2012). A Universal Push Service for Mobile Devices. *Sixth International Conference on Complex, Intelligent, and Software Intensive Systems* .

Estep, E. (2013). *Mobile HTML5: Efficiency and Performance of WebSockets and Server-Sent Events.* Sweden: Aalto University.

Guo, W., & Liu, H. (2013). The Analysis of Push Technology Based on Iphone Operating System. *2nd International Conference on Measurement, Information and Control* .

Pragya. (2016). Internet Based Communication using Server Push. *SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE)* .

Putra, A. Y. (2016). *Pengembangan Push Notification Menggunakan Websocket.* Malang: FILKOM Universitas Brawijaya.

Real-time Web Apps. (2017). Retrieved July 17, 2017, from blog.arkency.com: http://blog.arkency.com/2014/06/real-time-web-apps/

Skvorc, D., Horvat, M., & Srbljic, S. (2014). Performance Evaluation of Websocket Protocol for Implemetation of Full-Duplex Web Streams. *MIPRO* .

Webgeometrics.com. (n.d.). Retrieved July 15, 2017, from www.webgeometrics.com: http://www.webgeometrics.com/know-how-gcm-push-notifications-works-on-android-devices/

Zhang, L., & Shen, X. (2013). Research and Development of Real-time Monitoring System Based on WebSocket Technology. *International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC).*