Jurnal Sistim Informasi dan Teknologi

https://jsisfotek.org/index.php

2023 Vol. 5 No. 4 Hal: 63-72 e-ISSN: 2686-3154

Analisis Komparasi Protokol Websocket dan MQTT Dalam Proses Push Notification

Fauzan Prasetyo Eka Putra[1✉], Farhan Muslim[2], Nor Hasanah[3], Holipah[4],Reni Paradina[5], Royfal Alim[6]

[1,2,3,4,5,6]Universitas Madura

prasetyo@unira.ac.id

Abstrak

Teknologi push notification merupakan aspek krusial dalam aplikasi modern, yang memungkinkan pengiriman informasi secara instan kepada pengguna. Dua protokol komunikasi yang sering digunakan untuk mendukung fitur ini adalah Web Socket dan MQTT. Dalam penelitian ini, dilakukan analisis komparasi antara kedua protokol ini untuk memahami karakteristik, efisiensi, dan responsivitas masing-masing. Web Socket, dikenal karena koneksi dua arahnya (full-duplex), menyediakan kanal komunikasi persisten antara client dan server. Protokol ini memungkinkan pengiriman pesan secara real-time antara client dan server, sehingga sangat cocok untuk implementasi push notification yang membutuhkan responsivitas tinggi. Di sisi lain, MQTT (Message Queuing Telemetry Transport) dirancang khusus untuk keperluan Internet of Things (IoT). Fokus utama MQTT adalah pertukaran pesan terdistribusi, yang memungkinkan komunikasi efisien antara perangkat IoT. Penelitian ini menggunakan metode eksperimental dan pengukuran kinerja untuk mengevaluasi efisiensi dan responsivitas kedua protokol dalam mengirimkan push notification. Salah satu aspek yang diukur adalah kecepatan pengiriman pesan, yang melibatkan waktu delay dan waktu respons dari kedua protokol. Selain itu, penggunaan sumber daya seperti penggunaan CPU juga menjadi perhatian dalam penelitian ini. Hasil penelitian menunjukkan bahwa Web Socket cenderung lebih efisien dalam mengatasi push notification dalam skenario aplikasi real-time. Protokol ini menunjukkan waktu delay yang lebih rendah dan waktu respons yang lebih cepat dibandingkan dengan MQTT. Hal ini menunjukkan bahwa Web Socket mampu memberikan pengalaman pengguna yang lebih responsif dalam menerima push notification. Namun, MQTT menunjukkan keunggulan dalam pertukaran pesan terdistribusi pada lingkungan IoT. Protokol ini dirancang khusus untuk efisiensi dan kehandalan dalam komunikasi antara perangkat IoT. Dalam konteks penggunaan IoT, MQTT dapat memberikan performa yang lebih baik dalam mengirimkan push notification pada jaringan yang terdistribusi. Dengan demikian, kesimpulan dari penelitian ini adalah bahwa Web Socket lebih efisien dalam mengatasi push notification dalam skenario aplikasi real-time, sementara MQTT unggul dalam pertukaran pesan terdistribusi pada lingkungan IoT. Pemilihan protokol yang tepat harus didasarkan pada kebutuhan dan karakteristik aplikasi yang ingin diimplementasikan.

Kata kunci: Push Notification**,** Websocket, MQTT, Server, IoT

1. Pendahuluan

Dengan pesatnya perkembangan teknologi informasi, kebutuhan masyarakat terhadap akses informasi secara cepat dan berkala semakin meningkat. Untuk memenuhi kebutuhan ini, teknologi push notification menjadi solusi yang efektif dalam mengirimkan informasi secara otomatis dari server ke perangkat pengguna. Push notification merupakan teknologi yang memungkinkan pengiriman notifikasi ke perangkat pengguna tanpa memerlukan permintaan dari perangkat tersebut [1]. Dua protokol yang umum digunakan untuk implementasi push notification adalah Web Soket dan MQTT (Message Queuing Telemetry Transport).

Seiring dengan peningkatan kompleksitas aplikasi dan layanan berbasis web, pemilihan protokol untuk implementasi push notification menjadi faktor kritis. Web Soket adalah protokol yang mendukung komunikasi dua arah antara server dan client, sementara MQTT merupakan protokol pesan yang ringan dan efisien untuk komunikasi antar perangkat [2]. Keduanya memiliki karakteristik yang berbeda, dan penting untuk dilakukan analisis komparatif guna menentukan protokol mana yang lebih sesuai dalam konteks pengiriman notifikasi.

Sejumlah penelitian sebelumnya telah menginvestigasi performa protokol push notification, khususnya Web Soket dan MQTT. Elliot Estep [3] melakukan penelitian terkait efisiensi dan kinerja Web Soket dan Server-Sent Events (SSE) pada sisi client. Sementara itu, Muhammad et al. (2018) membandingkan kinerja protokol WebSocket dengan protokol SSE pada teknologi push notification [4]. Temuan mereka menunjukkan bahwa SSE memiliki rata-rata delay dan penggunaan CPU yang lebih rendah dibandingkan dengan WebSocket. Sebagai tambahan, peneliti lain mengembangkan mekanisme pengiriman notifikasi dengan protokol MQTT [5].

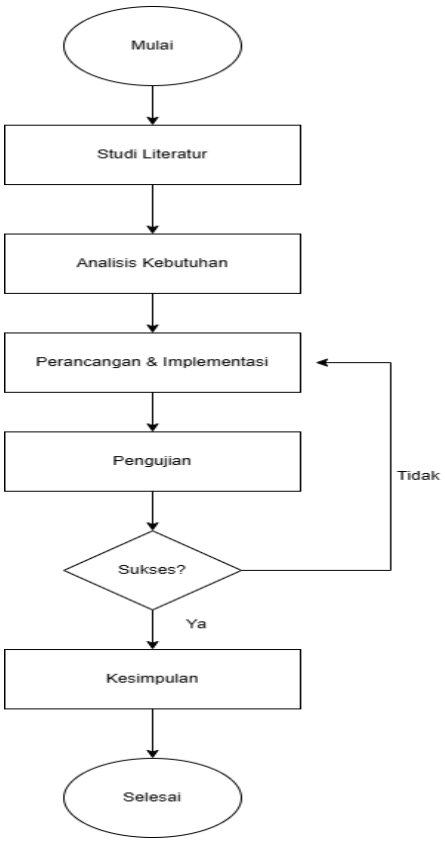
Meskipun penelitian sebelumnya telah memberikan wawasan yang berharga, masih diperlukan analisis lebih lanjut untuk memahami kinerja Web Soket dan MQTT dalam konteks aplikasi push notification. Alasan diadakannya

Diterima: 30-12-2023 | Revisi: 01-01-2024 | Diterbitkan: 08-01-2024 | doi: 10.60083/jsisfotek.v5i4.325

penelitian ini adalah untuk mengevaluasi dan membandingkan performa keduanya, khususnya dalam hal delay dan penggunaan CPU pada sisi server.
Dengan adanya perkembangan teknologi push notification dan pilihan protokol yang semakin beragam, analisis komparatif menjadi penting untuk menentukan solusi yang paling efektif. Penelitian ini diharapkan dapat memberikan kontribusi signifikan terhadap pemahaman tentang kinerja protokol Web Soket dan MQTT dalam konteks aplikasi push notification.

2. Metode Penelitian

Metodologi untuk merealisasikan penelitian ini melibatkan tahapan-tahapan yang dapat dilihat di Flow Chart Gambar 1.

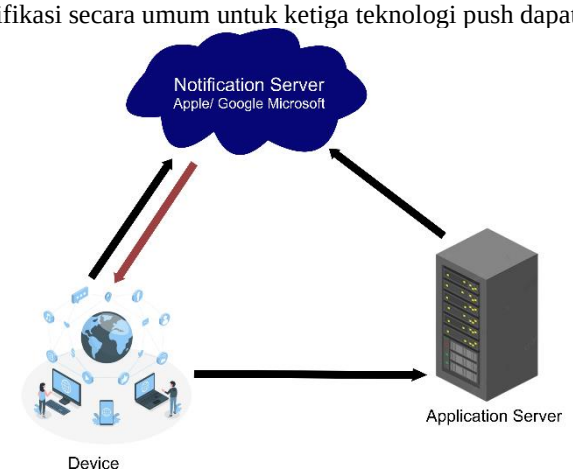


Gambar 1. Flow Chart Metodologi penelitian

Pertama, memulai dengan literature review adalah langkah yang penting untuk mengumpulkan informasi dan teori-teori yang relevan sebelum melanjutkan ke tahap analisis kebutuhan. Setelah itu, metode analisis kebutuhan membantu dalam menentukan perangkat lunak dan perangkat keras yang diperlukan untuk penelitian. Tahap implementasi melibatkan penerapan perangkat keras dan perangkat lunak yang telah dipilih. Terakhir, proses pengujian sistem dilakukan untuk memastikan bahwa sistem yang dikembangkan berfungsi dengan baik. Semua tahapan ini penting dalam merealisasikan penelitian yang efektif.

2.1. Pengertian push notification

Teknologi Push notification merujuk pada sistem pengiriman informasi dalam bentuk notifikasi dari server ke client tanpa memerlukan permintaan khusus dari client[23]. Ini memungkinan penerima, yaitu pengguna perangkat, untuk menerima informasi secara berkala dengan efisiensi tinggi. Umumnya, teknologi ini digunakan untuk mengirim informasi kepada pengguna smartphone [6]. Setiap sistem operasi smartphone memiliki pendekatan unik masing-masing terhadap Teknologi Push notification. Sistem operasi Android mengandalkan Google Cloud Messaging (GCM), sementara iPhone menggunakan Apple Push Notification Service (APNS) [7], dan Windows Phone mengandalkan Windows Phone Notification System (MPNS) [8]. Meskipun ada perbedaan dalam implementasi, ketiga teknologi ini memiliki kesamaan dengan menggunakan protokol Hypertext Transfer Protocol (HTTP). Protokol ini menjadi standar untuk mengatur pengiriman informasi di jaringan. Untuk memastikan bahwa notifikasi dari server dapat disampaikan dengan akurat ke penerima yang tepat, Pengguna smartphone perlu mendaftarkan ID perangkat mereka ke sistem push notificstion yang digunakan[23], [29]. Mekanisme pengiriman notifikasi secara umum untuk ketiga teknologi push dapat dilihat pada Gambar 2.



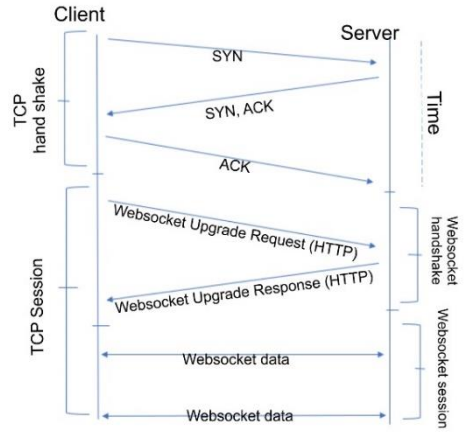
Gambar 2. Mekanisme Push Notification

2.2. Pengertian Protokol Websocket

WebSocket merupakan sebuah protokol yang dikembangkan dalam HTML5 [9]. Protokol ini memungkinkan komunikasi dua arah secara bersamaan (bidirectional full duplex) [10]. Hal ini berarti server dan client dapat mengirim dan menerima data secara bersamaan tanpa harus menunggu respons dari pihak lain [12]. WebSocket awalnya dirancang untuk digunakan di web browser dan web server, namun juga dapat digunakan oleh aplikasi client atau server lainnya. Protokol ini didasarkan pada protokol Transfer Control Protocol (TCP), yang merupakan protokol inti untuk mengatur pengiriman informasi di internet. Salah satu keunggulan utama WebSocket adalah kemampuannya untuk berbagi port dengan server HTTP [21]. Hal ini memungkinkan WebSocket untuk dengan mudah diterapkan dalam skala yang lebih besar.

Dalam hal fungsionalitas, WebSocket menyerupai TCP, namun dengan beberapa perbedaan [3]. WebSocket memiliki header yang lebih kecil dibandingkan dengan HTTP [25], yang membantu mengurangi overhead komunikasi. Namun, perlu dicatat bahwa WebSocket tidak dapat menggantikan TCP. WebSocket dan TCP bekerja pada lapisan jaringan yang berbeda dalam model protokol internet [11]. WebSocket memanfaatkan koneksi TCP yang sudah ada untuk melakukan komunikasi dua arah, tetapi tetap membutuhkan proses handshake agar client dapat terhubung dengan server.

WebSocket memungkinkan pengembang untuk membuat aplikasi web yang responsif dan interaktif dengan menghindari pembuatan koneksi baru setiap kali ada perubahan data yang perlu dikirim atau diterima. Dengan menggunakan WebSocket, aplikasi dapat mengirimkan pesan secara real-time antara server dan client, mengaktualisasi tampilan secara dinamis, dan memberikan pengalaman pengguna yang lebih baik.

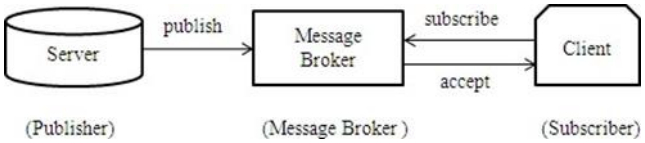


Gambar 3. Mekanisme kerja protokol Websocket

Pada gambar 3, memperlihatkan proses handshake antara client dan server dalam protokol Websocket. Untuk membuka koneksi Websocket, client mengirim permintaan koneksi ke server. Selanjutnya, server merespons permintaan koneksi dari client. Setelah proses handshake selesai, protokol Websocket memungkinkan pengiriman dan penerimaan data dalam model dual channel atau jalur yang berbeda secara bersamaan [10].
Server bertanggung jawab untuk mengirimkan notifikasi kepada client secara aktif. Proses pengiriman notifikasi akan terus berjalan sampai ada perintah untuk memutus koneksi [24]. Dalam protokol Websocket, pengiriman informasi dari server ke client tidak memerlukan pengenalan ulang client, sehingga ukuran paket yang dikirimkan lebih kecil dan efisien. Hal ini memungkinkan pengiriman informasi dengan kecepatan tinggi [11], [31]. Ketika koneksi antara client dan server terputus, server akan menerima suatu pemberitahuan [30]. Proses ini memastikan bahwa server dapat menangani keadaan ketika koneksi terputus dan mengambil tindakan yang sesuai.

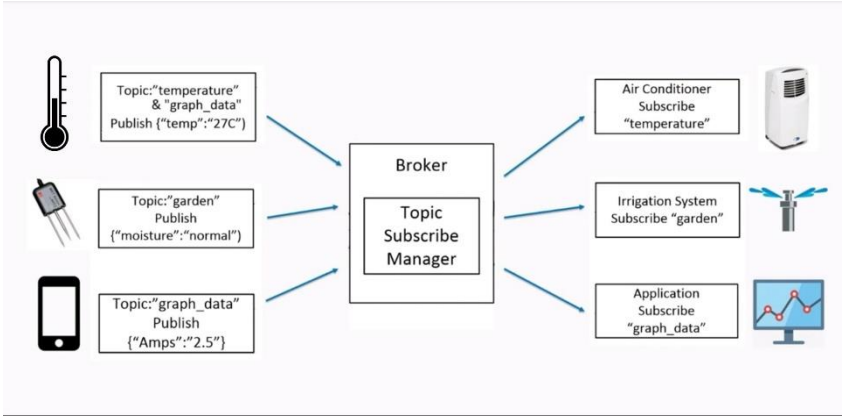
2.3. Pengertian Protokol MQTT

MQTT merupakan protokol pesan instan untuk komunikasi IoT(Internet of Things) dan aplikasi M2M (Machine-to-Machine)[22]. Juga merupakan protokol pesan berlangganan berbasis broker[20]. Fitur utamanya adalah bobotnya yang ringan. Metode komunikasi yang digunakan oleh protocol MQTT Adalah metode pengiriman publish/subscribe. Pesan yang diterima oleh MQTT akan dikirim ke broker dan berisi topik-topik yang dikirimkan oleh publisher. Kemudian topik-topik tadi diolah dan diteruskan ke subscriber berdasarkan dari permintaan pengguna. Protokol ini mendukung semua platform dan berbagai bahasa pemrograman populer. MQTT juga mendukung tiga tingkat Quality of Service (QoS) untuk memastikan kehandalan pengiriman pesan [14]. Kelebihan MQTT dalam pengiriman push notification meliputi pembaruan real-time, konsumsi daya yang rendah, efisiensi bandwidth, dan skalabilitas yang baik [15]. MQTT sangat cocok untuk aplikasi IoT yang membutuhkan notifikasi yang cepat dan andal. Karena kesederhanaan dan skalabilitasnya, banyak perusahaan domestik secara luas menggunakan protokol MQTT sebagai protokol push notification sisi server, seperti industri otomotif, industri manufaktur, logistik, komunikasi, minyak, dan gas. Menurut penciptanya, MQTT bahkan dapat diskalakan ke jutaan perangkat [13].



Gambar 4. Diagram Arsitektur Keseluruhan

Pada gambar 3, pengaturan saluran komunikasi MQTT cukup sederhana dan hanya membutuhkan dua partisipan, client, dan broker untuk memulai pengiriman pesan. Broker menyelesaikan fungsi perutean pesan, menyimpan dan menerima pesan yang dikirim oleh server(publisher), dan kemudian diteruskan. Clien(subscriber) berlangganan ke topik-topik yang menarik, untuk mengirimkan informasi ke Message Broker, dan untuk mempertahankan koneksi secara real-time. Salah satu contoh pada gambar 5, sebuah sensor suhu mempublikasikan data suhu ke dalam jaringan. Lalu clien berlangganan(subscribe) ke Suhu(temperature) [13], [28].



Gambar 6. Model publish/subscribe MQTT

2.4. Perancangan dan Implementasi

Penelitian ini bertujuan untuk membandingkan implementasi protokol Websocket dan protokol MQTT dalam mekanisme pengiriman push notification pada jaringan lokal yang sama (WLAN). Dalam penelitian ini, dilakukan pengujian terhadap protokol Websocket dan protocol Mqtt untuk melihat performa dan efisiensi keduanya dalam mengirimkan push notification.

Tabel 1. Kebutuhan Server dan Client

	Server	***Client***
	Laptop	***Smartphone***
Bahasa Pemrograman	Python	Javascript
Framework	Flask	-
Library	Socket.IO	-
Sistem Operasi	Windows 11	Android

Dalam tabel 1, terlihat bahwa perangkat yang berperan sebagai server adalah sebuah laptop yang menjalankan sistem operasi Windows 11. Dalam implementasi pengiriman notifikasi, digunakan teknologi Websocket dan MQTT pada sisi server dengan menggunakan bahasa pemrograman Python [18]. Selain itu, untuk mendukung pengembangan server, digunakan framework Flask[26] dan library Socket.IO[27]. Sementara itu, perangkat yang berperan sebagai client adalah sebuah smartphone yang menggunakan sistem operasi Android. Bahasa pemrograman yang digunakan untuk merancang halaman situs yang akan diakses oleh client melalui browser adalah Javascript. Dalam konteks ini, Javascript berperan penting dalam merancang tampilan halaman situs yang interaktif dan responsif bagi pengguna smartphone [17].

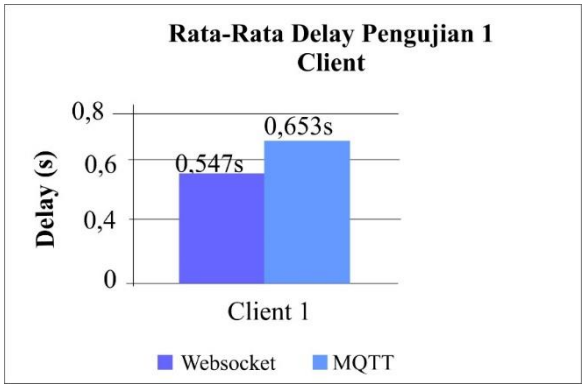
3. Hasil dan Pembahasan

Pengujian dilakukan dengan mengirimkan notifikasi kepada 1 client, 2 client, dan 5 client. Setelah itu, hasil dari ketiga skenario tersebut akan dianalisis untuk mendapatkan rata-rata delay pada masing-masing skenario. Selain itu, pengujian juga melibatkan monitoring resource dengan menggunakan modul psutil. Modul ini akan memberikan data penggunaan CPU dalam bentuk log pada server. Untuk mendapatkan data penggunaan CPU, dilakukan pengurangan nilai CPU setelah pengiriman notifikasi dengan nilai CPU sebelum pengiriman. Hasil penggunaan CPU akan ditampilkan dalam bentuk persentase di sisi server.

Dengan melakukan pengujian ini, kita dapat memperoleh informasi yang penting mengenai performa jaringan dan penggunaan sumber daya. Rata-rata delay pada setiap skenario akan memberikan gambaran tentang waktu yang dibutuhkan untuk mengirim notifikasi kepada jumlah client yang berbeda. Sementara itu, data penggunaan CPU akan memberikan informasi tentang seberapa besar pengaruh pengiriman notifikasi terhadap kinerja CPU pada server

3.1. Pengujian Pengiriman Pada 1 Client

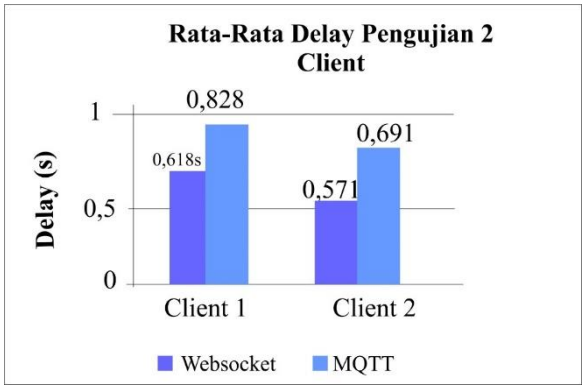
Gambar 7 menunjukan data rata-rata delay yang dihasilkan dari pengujian menggunakan satu client uji. Berdasarkan grafik yang dihasilkan dari data ini, tergambar jelas bahwa terdapat perbedaan yang signifikan dalam waktu delay antara pengiriman data melalui protokol Websocket dan MQTT. Hasil pengujian menunjukkan bahwa Websocket memiliki waktu delay sebesar 0.547s(second/detik), sementara MQTT mencapai 0.653s. Selisih delay keduanya yaitu 0.106.



Gambar 7. Rata-Rata Delay Pengujian 1 Client

3.2. Pengujian pengiriman pada 2 client

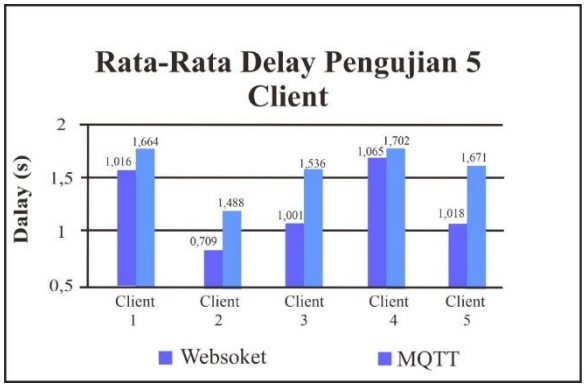
Berikut adalah grafik rata-rata delay dalam proses pengiriman notifikasi dengan menggunakan 2 client.



Gambar 8. Rata-Rata Delay Pengujian 2 Client

Gambar 8 menunjukan hasil pengujian kinerja jaringan menggunakan dua client, terdapat perbedaan waktu respons antara protokol Websocket dan MQTT. Websocket menunjukkan kinerja lebih baik dengan rata-rata waktu respons sekitar 0.595s, sementara MQTT memiliki rata-rata waktu respons sekitar 0.759s.

3.3. Pengujian Pengiriman Pada 5 Client

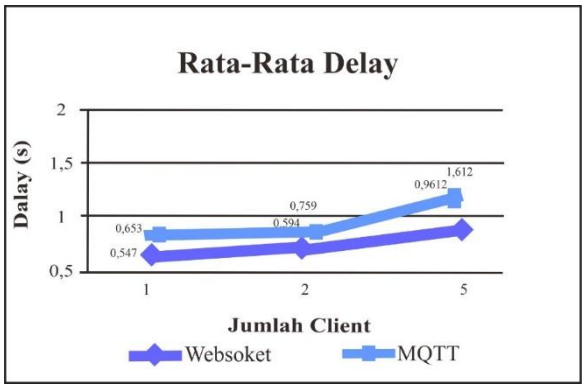


Gambar 9. Rata-Rata Delay Pengujian 5 Client

Selanjutnya adalah pengujian pengiriman notifikasi menggunakan 5 client. Pada gambar 9 dapat dilihat perbedaan delay yang sangat besar dibandingkan dengan pengujian sebelumnya menggunakan 1 client dan 2 client. Di gambar

10 pengukuran waktu respons protokol Websocket dan MQTT pada lima client memberikan gambaran yang menarik. Dalam pengujian ini, dapat diamati bahwa Websocket secara konsisten memberikan waktu respons yang lebih rendah dibandingkan dengan MQTT. Rata-rata waktu respons Websocket pada kelima client adalah sekitar 1.004s, sedangkan MQTT memiliki rata-rata waktu respons sekitar 1.614s. Perbedaan ini menunjukkan bahwa Websocket memiliki performa yang lebih baik dalam hal respons dan transmisi data dibandingkan dengan MQTT dalam konteks pengujian push notification

3.4 Analisis Pengujian Push Notification

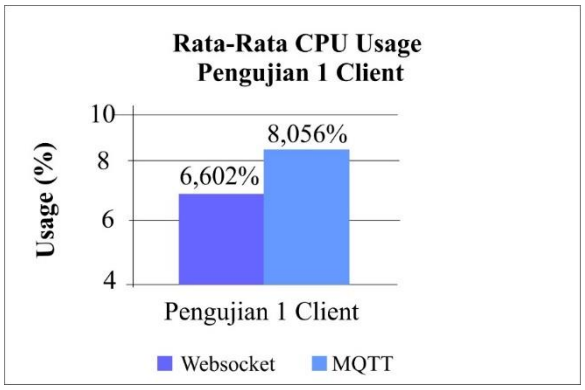


Gambar 10. Rata-Rata Delay

Gambar 10 Pertambahan jumlah client sangat mempengaruhi pertambahan nilai rata-rata delay saat proses pengiriman pesan. Pengujian dengan 1 client pada Protocol Websocket menghasilkan total delay sekitar 0.547s, sedangkan MQTT memiliki total delay sekitar 0.653s. Perbandingan ini menunjukkan bahwa Websocket memperoleh kinerja yang lebih baik dalam pengiriman notifikasi. Pada pengujian dengan 2 client, client pertama menggunakan Websocket dengan total delay sekitar 0.618s, sedangkan dengan MQTT total delay adalah 0.828s. Client kedua menggunakan Websocket dengan total delay sekitar 0.571s, dan dengan MQTT total delay adalah 0.691s. Walaupun perbedaan tidak terlalu besar, Websocket tetap menunjukkan kinerja lebih baik pada kedua skenario pengujian dengan dua client. Pengujian dengan 5 client menunjukan client pertama menggunakan Websocket dengan total delay sekitar 1.016s, sementara dengan MQTT total delay adalah 1.664s. Demikian pula, client-client lainnya menunjukkan bahwa Websocket memiliki total delay yang lebih rendah dibandingkan dengan MQTT.

Dapat diamati bahwa rata-rata total delay Websocket lebih rendah dibandingkan dengan MQTT pada setiap skenario pengujian. Hal ini dikarenakan Websocket menyediakan koneksi yang persisten antara client dan server, mengizinkan komunikasi dua arah tanpa harus membuka koneksi baru setiap kali ada data yang dikirim. Sebaliknya, MQTT memerlukan pembukaan dan penutupan koneksi setiap kali ada interaksi, yang dapat menambah overhead. Dan juga Websocket dirancang khusus untuk komunikasi real-time dan streaming data, membuatnya lebih efisien dalam konteks aplikasi yang membutuhkan respons cepat.

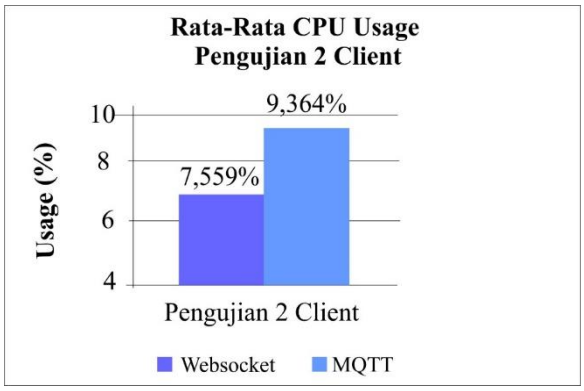
3.5 Pengujian Monitoring Resource Pada 1 Client



Gambar 11. Rata-Rata CPU Usage Pengujian 1 Client

Dari grafik pada gambar 11, dapat dilihat bahwa hasil pengujian protokol Websocket menunjukkan penggunaan CPU sebesar 6.602%, sedangkan MQTT sebesar 8.056%. Data ini mengindikasikan bahwa Websocket memiliki efisiensi CPU yang sedikit lebih baik dibandingkan MQTT pada pengujian satu client.

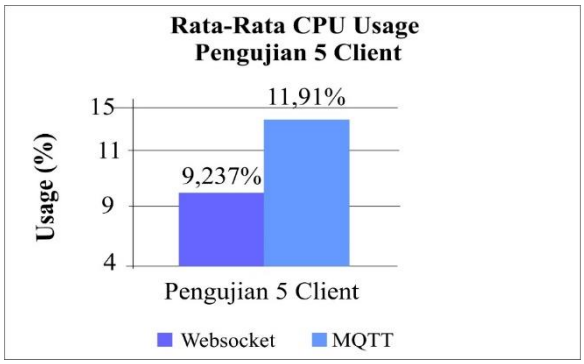
3.6 Pengujian Monitoring Resource Pada 2 Client



Gambar 12. Rata-Rata CPU Usage Pengujian 2 Client

Berdasarkan gambar grafik pada gambar 12, Hasil pengujiannya menunjukkan bahwa pada dua client, Websocket memiliki rata-rata penggunaan CPU sebesar 7.559%, sementara MQTT sebesar 9.364%. Websocket tetap menunjukkan efisiensi CPU yang lebih baik dibandingkan MQTT, informasi ini dapat membantu dalam pemilihan protokol komunikasi berdasarkan performa CPU.

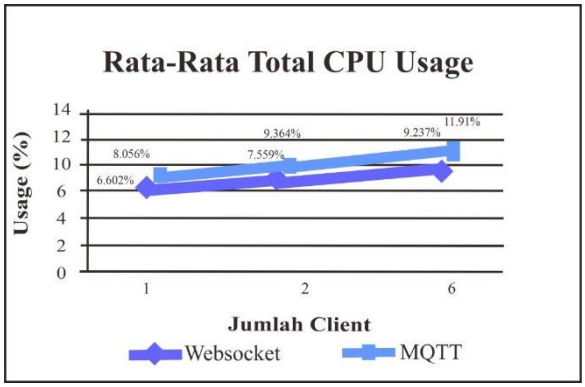
3.7 Pengujian Monitoring Resource Pada 5 Client



Gambar 13. Rata-Rata CPU Usage Pengujian 5 Client

Pada grafik pada gambar 13, Websocket menunjukkan rata-rata penggunaan CPU sebesar 9.237%, sedangkan MQTT mencapai 11.91%. Perbandingan ini menunjukkan bahwa Websocket masih memiliki efisiensi CPU lebih baik dibandingkan MQTT pada situasi pengujian dengan jumlah client yang lebih besar.

3.8 Analisis Pengujian Monitoring Resource



Gambal 14 Rata-Rata Total CPU Usage

Pada gambar 14 hasil pengujian CPU usage dengan jumlah client yang berbeda, terlihat adanya perbedaan kenaikan penggunaan CPU antara protokol Websocket dan MQTT. Pada pengujian dengan satu client, Websocket menunjukkan CPU usage sebesar 6.602%, sedangkan MQTT memiliki CPU usage 8.056%. Saat jumlah client meningkat menjadi dua, Websocket mengalami kenaikan CPU usage menjadi 7.559%, sementara MQTT meningkat menjadi 9.364%. Perbedaan ini semakin mencolok saat pengujian dilakukan dengan lima client, di

mana Websocket menunjukkan CPU usage sebesar 9.237%, sedangkan MQTT mengalami kenaikan yang lebih signifikan menjadi 11.91%.
Analisis data menunjukkan bahwa semakin banyak jumlah client, MQTT cenderung memiliki kenaikan CPU usage yang lebih tinggi dibandingkan dengan Websocket. Perbedaan ini dapat dipengaruhi oleh karakteristik masing-masing protokol dalam menangani koneksi dan pertukaran data. Websocket, dengan tingkat kenaikan yang lebih moderat, mungkin lebih efisien dalam penggunaan sumber daya CPU dibandingkan dengan MQTT saat menghadapi beban koneksi yang lebih besar. Namun, perlu dicatat bahwa faktor lain seperti konfigurasi sistem dan implementasi spesifik dari masing-masing protokol juga dapat memengaruhi hasil pengujian ini.

4. Kesimpulan

Berdasarkan hasil pengujian, dapat disimpulkan bahwa penggunaan protokol Websocket memberikan kinerja yang lebih unggul dari pada protokol MQTT dalam proses push notification. Dalam pengujian tersebut, terlihat bahwa rata-rata total delay keseluruhan dari client uji pada protokol Websocket adalah 0.700s, sedangkan pada protokol MQTT adalah 1.008s. Selain itu, penting untuk dicatat bahwa jumlah client uji juga berpengaruh terhadap delay yang diperoleh. Semakin banyak client yang menerima notifikasi, maka delay akan semakin meningkat. Ini menunjukkan bahwa protokol Websocket menawarkan kinerja yang lebih baik dalam hal delay dibandingkan dengan protokol MQTT.
Selain itu, dalam pengujian penggunaan CPU, Websocket juga menunjukkan efisiensi yang lebih baik daripada MQTT. Pada semua skenario pengujian, Websocket memiliki rata-rata penggunaan CPU yang lebih rendah dibandingkan dengan MQTT. Pada pengujian dengan 1 client, Websocket memiliki rata-rata penggunaan CPU sebesar 6.602%, sedangkan MQTT memiliki rata-rata penggunaan CPU sebesar 8.056%. Saat jumlah client bertambah hingga lima, Websocket tetap terus menunjukkan efisiensi dengan perbedaan penggunaan CPU yang mencapai 2.673%.
Meskipun hasil penelitian menunjukkan performa Websocket lebih unggul dalam hal total delay dan efisiensi CPU, kegunaan MQTT masih tetap sangat signifikan. MQTT, sebagai protokol komunikasi, memiliki keunggulan di berbagai skenario, terutama pada implementasi IoT dan M2M. MQTT sering digunakan dalam situasi di mana delay dan penggunaan CPU bukan faktor penentu utama, namun, kehandalan dan efisiensi penggunaan sumber daya jauh lebih penting [16].

Daftar Rujukan

[1] Hansen, M., et al. (2012). "Push Notification Systems: A Survey." IEEE Communications Surveys & Tutorials.
[2] Zhang, X., & Shen, H. (2013). "Research on Web Socket Protocol and Its Application in Communication Systems." 2013 Ninth International Conference on Computational Intelligence and Security.
[3] Estep, E. (2013). "Mobile HTML5: Efficiency and Performance of Websockets and Server-Sent Events."
[4] Muhammad, et al. (2018). "Performance Comparison of WebSocket and Server-Sent Events for Push Notification Technology." Journal of Physics: Conference Series, 978(1).
[5] Konglong Tang et al. (2013). " Design and Implementation of Push Notification System Based on the MQTT Protocol "
[6] Amazon, 2023. Amazon Simple Notification Service. [Online] Available at: https://docs.aws.amazon.com/sns/latest/dg/welcome.html
[7] Apple Inc., 2023. Registering your app with APNs [Online] Available at: https://developer.apple.com/documentation/usernotifications/registering_your_app_with_apns
[8] Brüstel, Jonas. & Preuss, Thomas., 2012. A Universal Push Service for Mobile Devices. 2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems.
[9] Hickson, Ian. & et., a., 2014. HTML5: A Vocabulary and Associated APIs for HTML and XHTML. [Online] Available at: http://www.w3.org/TR/html5
[10] 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.
[11] Skvorc, D., Horvat, M. & Srbljic, S., 2014. Performance Evaluation of Websocket Protocol for Implemetation of Full-Duplex Web Streams. MIPRO.
[12] Mozilla Corporation. 2022. API WebSocket (WebSockets). Diakses 29.3.2022. https://developer.mozilla.org/en-US/docs/Web/API/WebSock- ets_API
[13] mqtt.org. 2022. MQTT: The Standard for IoT Messaging [Online] Available at: https://mqtt.org/
[14] Yokotani, T. & Sasaki, Y. 2016. Comparison with HTTP and MQTT on required network resources for IoT.
[15] Prasetyo, Hariyo. & dkk. 2023. Analisis Perbandingan Kinerja Protokol MQTT dan AMQP pada Pengiriman Pesan Peringatan dari Sensor NIDS berdasarkan Serangan Jaringan.
[16] Enache, B. A., Banica, C. K., & Bogdan, A. G. (2023). Performance Analysis of MQTT Over Websocket for IoT Applications. The Scientific Bulletin of Electrical Engineering Faculty, 23(1), 46–49. https://doi.org/10.2478/sbeef-2023-0008
[17] Yudianto, A., Sakti, E., Kom, S., Kom, M., & Amron, K. (2017). PENGEMBANGAN PUSH NOTIFICATION MENGGUNAKAN WEBSOCKET. In *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK* (Vol. 1, Issue 1).
[18] Budi, R., 2015. Mudah Belajar Python Untuk Aplikasi dan Web. Bandung: Informatika.

[19] Naik, N. 2017. Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP.
[20] Tang, K., Wang, Y., Liu, H., Sheng, Y., Wang, X., & Wei, Z. (2013). Design and Implementation of Push Notification System Based on the MQTT Protocol. https://doi.org/10.2991/isca-13.2013.20
[21] Soewito, B., Christian, Gunawan, F. E., Diana, & Gede Putra Kusuma, I. (2019). Websocket to support real time smart home applications. Procedia Computer Science, 157, 560–566. https://doi.org/10.1016/j.procs.2019.09.014
[22] Mishra, B., & Kertesz, A. (2020). The use of MQTT in M2M and IoT systems: A survey. IEEE Access, 8, 201071–201086. https://doi.org/10.1109/ACCESS.2020.3035849
[23] Tommy, L., Wahyuningsih, D., & Romadiana, P. (2020). Pengembangan Aplikasi Penerimaan Mahasiswa Baru Berbasis Android dengan Push Notification di STMIK Atma Luhur. Jurnal Sisfokom (Sistem Informasi Dan Komputer), 9(1), 108–121. https://doi.org/10.32736/sisfokom.v9i1.813
[24] Guan, S., Hu, W., & Zhou, H. (n.d.). Real-Time Data Transmission Method Based on Websocket Protocol for Networked Control System Laboratory. https://www.powersim.whu.edu.cn/ncslab.
[25] Łasocha, W. P., & Badurowicz, M. (2021). Comparison of WebSocket and HTTP protocol performance Porównanie wydajności protokołu WebSocket i HTTP. In JCSI (Vol. 19).
[26] Ningtyas, D. F., & Setiyawati, N. (2021). Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request. Jurnal Janitra Informatika Dan Sistem Informasi, 1(1), 19–34. https://doi.org/10.25008/janitra.v1i1.120
[27] Setyawan, S., & Susanto, A. (2018). Sistem Push Notification Life@BankJateng Sebagai Penunjang Layanan Pegawai PT. Bank Jawa Tengah Menggunakan Socket.IO Push Notification System Life@BankJateng to Support PT. Bank Jawa Tengah Using Socket.IO. Journal of Information System, 03.
[28] Sameer Sadeq, A., Sleibi Al-rawi, S., Mahdi Jubair, A., & Hafizah Mohd Aman, A. (2019). A QOS APPROACH FOR INTERNET OFTHINGS (IOT) ENVIRONMENT USING MQTT PROTOCOL.
[29] Yunisa, Fiona., & Suharijto. (2016). Push Notification System to Mobile Game Player Using Distributed Event-Based System Approach, d International Conference on Science in Information Technology (ICSITech).
[30] Hu, Yangyang., & Chen, Weiqing. (2017). Research and Implementation of Campus Information Push System Based on WebSocket, International Conference on Intelligent Systems and Knowledge Engineering (ISKE)
[31] Rahmatulloh, A., dkk. (2019). Performance Analysis of Data Transmission on WebSocket for Real-time Communication. https://ieeexplore.ieee.org/document/8898135