Jurnal Infomedia : Teknik Informatika, Multimedia, dan Jaringan
Volume 10, Number 1, Juni 2025
ISSN 2527-9858 (online), 2549-1180 (print)
Artikel Penelitian

Perbandingan Kinerja Protokol MQTT dan HTTP Dalam Komunikasi Data *Internet of Things*

Aditya Aziz Fikhri*[1] Munirul Ula[2] Muhammad Sayuti[3] Taufiq[4] Nurdin[5]
[1] Program Studi Magister Teknologi Informasi, Universitas Malikussaleh, Bukit Indah, Lhokseumawe, 24355, Indonesia, aditfreedom11@gmail.com
[2] Program Studi Magister Teknologi Informasi, Universitas Malikussaleh, Bukit Indah, Lhokseumawe, 24355, Indonesia, munirulula@unimal.ac.id
[3] Industrial Engineering Departement, Engineering Faculty, Universitas Malikussaleh, Indonesia, 24355, sayuti_m@unimal.ac.id
[4] Program Studi Magister Teknologi Informasi, Universitas Malikussaleh, Bukit Indah, Lhokseumawe, 24355, Indonesia, taufiq.te@unimal.ac.id
[5] Program Studi Magister Teknologi Informasi, Universitas Malikussaleh, Bukit Indah, Lhokseumawe, 24355, Indonesia, nurdin@unimal.ac.id
*Corresponding Author: **aditfreedom11@gmail.com (081362059403)**

Abstrak
Penelitian ini membandingkan kinerja protokol MQTT dan HTTP dalam sistem komunikasi Internet of Things (IoT), khususnya untuk pemantauan kualitas udara ruang kelas secara real-time. Evaluasi dilakukan menggunakan server virtual machine dengan spesifikasi identik, berdasarkan parameter seperti penggunaan CPU, waktu pengiriman pesan, dan tingkat kehilangan data. MQTT, sebagai protokol ringan dengan model publish-subscribe, menunjukkan kecepatan pengiriman pesan yang jauh lebih tinggi dibandingkan HTTP, terutama pada skenario dengan volume pesan yang besar. Namun, penggunaan CPU pada MQTT meningkat tajam seiring bertambahnya jumlah pesan, dan terjadi kehilangan data yang signifikan hingga 33,8% pada pengiriman 600.000 pesan. Sebaliknya, HTTP yang berbasis model request-response dengan mekanisme multi-proses, mampu menjaga keandalan pengiriman pesan hingga 100%, meskipun waktu pengirimannya jauh lebih lambat. Hasil penelitian ini menunjukkan bahwa MQTT lebih efisien untuk sistem yang membutuhkan kecepatan tinggi dan dapat mentoleransi sebagian kehilangan data, sementara HTTP lebih cocok untuk aplikasi yang menuntut keandalan tinggi dan akurasi data secara penuh. Temuan ini memberikan wawasan penting bagi pengembang dalam memilih protokol komunikasi yang sesuai berdasarkan kebutuhan sistem IoT dan skala implementasinya.
Keywords: Internet of Things, MQTT, HTTP, Autocannon, MQTT Stresser.

Abstract
This study compares the performance of MQTT and HTTP protocols in Internet of Things (IoT) communication systems, particularly for real-time classroom air quality monitoring. The research evaluates both protocols through identical virtual machine servers using parameters such as CPU usage, message delivery time, and data loss rates. MQTT, a lightweight protocol using the publish-subscribe model, demonstrated superior speed in message transmission, significantly outperforming HTTP in scenarios with high message volumes. However, MQTT's CPU usage increased drastically with the number of messages, and it experienced substantial message loss, reaching up to 33.8% at 600,000 messages. In contrast, HTTP, which operates on a request-response model with a multi-process mechanism, maintained 100% message delivery reliability, although with considerably longer transmission times. The results indicate that MQTT is more efficient for systems requiring rapid data transfer and can tolerate some data loss, while HTTP is preferable for applications that demand high reliability and complete data accuracy. The findings offer valuable insights for developers when selecting appropriate communication protocols based on specific IoT system requirements and scalability considerations.
Keywords: Internet of Things, MQTT, HTTP, Autocannon, MQTT Stresser.

PENDAHULUAN

Dalam era kemajuan sistem otomasi saat ini, manajemen perangkat keras semakin mengandalkan sistem tertanam (*embedded system*) yang didukung oleh perkembangan teknologi komunikasi dan mikrokontroler (Oliveira et al., 2024). Seiring dengan itu, pengembangan aplikasi berbasis jaringan menghadirkan tantangan tersendiri pada berbagai industri, dan dituntut untuk membuat sistem manajemen dan pemantauan yang dapat beroperasi secara tersentralisasi maupun terdesentralisasi yang berguna untuk meningkatkan efisiensi, fleksibilitas, dan keandalan dalam pengambilan keputusan serta respons terhadap kondisi di lapangan secara *real-time* (Arshad et al., 2023).

Transformasi ini dimulai dengan diperkenalkannya konsep Industri 4.0 (Nikolov, 2020). Salah satu konsep yang dikembangkan pada Industri 4.0 yaitu *Internet of Things* (IoT) (Kumar et al., 2022; Malik et al., 2021), IoT merupakan konsep di mana berbagai perangkat fisik seperti sensor, alat elektronik, kendaraan, peralatan rumah tangga, dan lainnya terhubung ke internet dan dapat saling bertukar data (Kopetz & Steiner, 2022; Yunana et al., 2021). Berbagai standar dan protokol komunikasi telah dikembangkan untuk IoT untuk memastikan konektivitas, efisiensi, dan keandalan sistem. Dua protokol komunikasi yang banyak digunakan dalam IoT adalah MQTT (*Message Queue Telemetry Transport*) dan HTTP (*Hypertext Transfer Protocol*) (Mishra & Kertesz, 2020; Nugraha et al., 2024).

MQTT adalah protokol berbasis *publish-subscribe* yang ringan dengan mengandalkan efisiensi *bandwidth* tinggi, yang mana ideal untuk aplikasi yang membutuhkan komunikasi data *real-time* dengan latensi rendah dan konsumsi daya yang sedikit (Jara Ochoa et al., 2023; Kadam & Ghule, 2023). Di sisi lain, HTTP merupakan protokol berbasis *request-response* yang lebih umum digunakan dalam aplikasi berbasis web karena kemampuannya dalam mentransmisikan data berbasis teks, gambar, dan lainnya yang lebih baik dari MQTT. Namun, protokol HTTP memiliki *overhead* yang lebih tinggi dibandingkan dengan MQTT (Jara Ochoa et al., 2023). Untuk kinerja yang optimal, perangkat IoT harus beroperasi dalam lingkungan tertentu dan berkomunikasi secara efisien satu sama lain saat mengumpulkan dan mengirimkan data ke *cloud*. Efisiensi pertukaran data bergantung pada karakteristik protokol komunikasi yang digunakan, termasuk faktor-faktor seperti latensi, keandalan, keamanan, dan efisiensi daya (Zimmerling et al., 2021). Oleh karena itu, pemilihan protokol komunikasi yang tepat sangat penting dalam mengembangkan sistem berbasis IoT, khususnya untuk aplikasi yang memerlukan pemantauan data secara *real-time*.

Saat ini, sistem IoT untuk pemantauan kualitas udara pada ruang kelas sudah diimplementasikan dan beroperasi dengan baik di salah satu ruang kelas di Sekolah Sukma Bangsa Bireuen, Provinsi Aceh, Indonesia. Hal ini dilakukan untuk merespon isu perubahan iklim yang terjadi di seluruh dunia serta mengevaluasi tingkat kenyamanan peserta didik yang dilihat dari kualitas udara agar proses pembelajaran akan menjadi semakin baik dan pelayanan sekolah menjadi semakin berkualitas. Adapun parameter kualitas udara yang dipantau meliputi suhu, kelembapan, CO, CO_2, PM2.5, dan PM10 dengan menggunakan mikrokontroller ESP32, sensor DHT22, MQ-9, MQ-135, dan DSM510A.

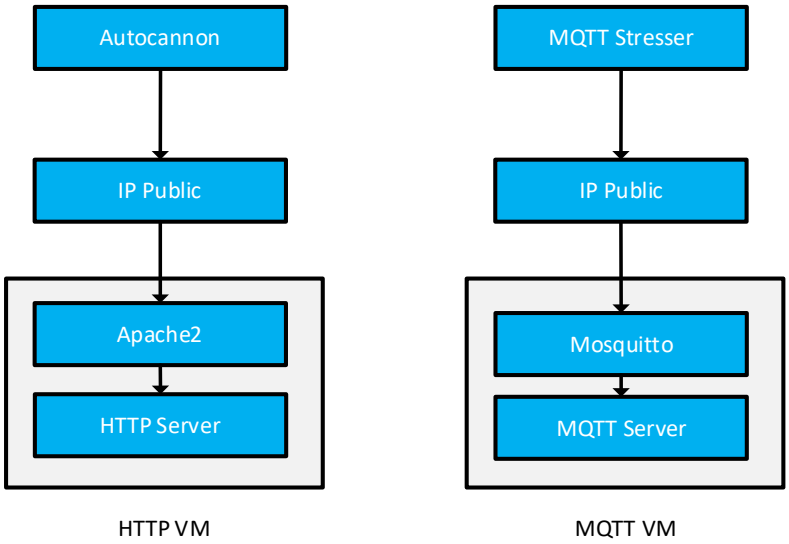
Namun pada kenyataannya, sebuah sekolah umumnya memiliki banyak ruang kelas yang masing-masing memerlukan pemantauan kualitas udara juga. Tantangan semakin besar jika sistem diperluas ke beberapa sekolah sekaligus, yang berarti jumlah sensor, mikrokontroler, dan perangkat yang harus dikelola secara terpusat juga meningkat secara signifikan. Dalam skenario seperti ini, beban data yang masuk ke server menjadi sangat besar dan berpotensi menyebabkan keterlambatan dalam pengiriman serta pemrosesan data (*delay*). Oleh karena itu, dalam implementasi sistem IoT dalam skala besar, selain merancang arsitektur sistem yang tepat, diperlukan juga pemilihan protokol komunikasi yang tidak hanya andal dan efisien, tetapi juga mampu menangani lalu lintas data dalam jumlah besar tanpa mengorbankan kecepatan dan akurasi informasi agar membuat server bekerja lebih optimal (Khalifeh et al., 2022). MQTT dan HTTP dalam hal ini merupakan dua protokol komunikasi yang umum digunakan dalam sistem IoT (Mishra & Kertesz, 2020; Nugraha et al., 2024), namun keduanya memiliki karakteristik teknis yang berbeda dalam hal efisiensi pengiriman data, penggunaan *bandwidth*, dan kemampuan *scaling* (Alshammari, 2023; Silva et al., 2021).

Maka dari itu, penelitian ini bertujuan untuk menganalisis secara komprehensif perbandingan kinerja antara protokol MQTT dan HTTP dengan menggunakan spesifikasi perangkat keras server pengujian yang identik. Analisis ini mencakup aspek penggunaan CPU pada server terhadap jumlah *client* dan pesan/data, kecepatan waktu pengiriman data, dan tingkat penerimaan atau kehilangan data (*data loss*). Hasil dari penelitian ini diharapkan dapat memberikan rekomendasi mengenai protokol komunikasi yang paling optimal untuk diterapkan dalam sistem pemantauan kualitas udara berbasis IoT di ruang kelas, khususnya pada lingkungan yang membutuhkan pemantauan *real-time* secara simultan di banyak titik.

METODE PENELITIAN

A. Arsitektur Sistem

Perancangan arsitektur sistem dilakukan untuk menentukan perangkat lunak dan keras yang digunakan pada setiap protokol, arsitektur sistem pengujian dapat dilihat pada Gambar 1.



Gambar 1. Arsitektur Sistem

Untuk mengaktifkan server menjadi publik, setiap VM diberi alamat IP publik. Selain itu, mekanisme operasional protokol HTTP dan MQTT berbeda yang mana HTTP mengikuti model *request-response* (Asyhari et al., 2019), sementara MQTT menggunakan mekanisme publish-subscribe (Mishra & Kertesz, 2020). Karena perbedaan ini, terdapat pula perangkat lunak pengujian yang berbeda, pada HTTP menggunakan Autocannon yang menghasilkan beberapa *request* ke server HTTP (Challapalli et al., 2021), sedangkan MQTT menggunakan MQTT Stresser yang mengirimkan sejumlah besar pesan ke broker MQTT lokal (Mishra, 2018).

Tabel 1. Spesifikasi VM

Properti	Nilai
Sistem Operasi	Ubuntu 22.04 LTS
vCPU	1 Core
RAM	6 GB
vSSD	60 GB

Dalam penelitian ini, protokol MQTT diatur untuk beroperasi pada Quality of Service (QoS) level 0, yang berarti pengirim tidak menerima konfirmasi apapun terkait pengiriman pesan yang mana QoS 0 pada protokol MQTT sejalan dengan sifat HTTP yang mana jika pengiriman data pada saat *request* terjadi, maka pengguna harus mengirim ulang data tersebut, sehingga protokol HTTP juga tidak memiliki mekanisme mengiriman ulang data yang dikirimkan secara otomatis jika data gagal dikirimkan.

B. Definisi Jumlah Pesan dan *Client*

Dalam menentukan jumlah pesan, terdapat perbedaan antara Autocannon dan MQTT Stresser. Pada Autocannon, jumlah pesan dapat ditentukan secara langsung, sedangkan pada MQTT Stresser jumlah pesan harus diatur secara individual untuk setiap *client*. Selain itu, Autocannon dapat dilakukan konfigurasi beberapa pengiriman secara bersamaan dengan jumlah pesan yang telah ditentukan, sehingga memudahkan dalam mengendalikan total beban data. Sebaliknya, MQTT Stresser mengharuskan konfigurasi terpisah untuk setiap *client*, yang memberikan fleksibilitas namun dapat menjadi lebih kompleks ketika diuji dalam skala besar. Penjelasan lebih rinci mengenai hal ini disajikan pada Tabel 2.

Tabel 2. Pembulatan Pada MQTT Stresser

Client	Pesan	Pesan Per *Client*	Pembulatan
500	15.000	30	15.000
1.000	15.000	15	15.000
1.500	15.000	10	15.000
2.000	15.000	8	16.000

Pada MQTT Stresser, jumlah pesan per *client* diperoleh dengan membagi total pesan dengan jumlah *client*, sebagaimana dijelaskan pada Persamaan (1).

$$\text{Pesan per } client = \frac{\text{Pesan}}{Client} \quad (1)$$

Dengan menggunakan Persamaan (1), pembagian dapat menghasilkan nilai desimal, sehingga dibutuhkan pembulatan nilai menjadi bilangan bulat terdekat. Proses pembulatan memastikan bahwa setiap *client* menerima distribusi pesan yang sama, mencegah nilai pecahan yang dapat menciptakan ketidakseimbangan dalam kondisi pengujian. Misalnya, jika 15.000 pesan didistribusikan di antara 2.000 *client*, hasilnya adalah 7,5. Dalam hal ini, nilainya akan dibulatkan menjadi 8. Untuk menentukan nilai total pesan hasil pembulatan didapatkan dari pesan per *client* dikalikan dengan jumlah total *client*, seperti yang dijelaskan dalam Persamaan (2).

$$\text{Total Hasil Pembulatan} = \text{Pesan per } client \text{ x jlh. } client \quad (2)$$

Variasi antara jumlah pesan awal dan jumlah pesan hasil pembulatan muncul sebagai akibat dari proses pembulatan. Misalnya, dalam pengujian di mana 15.000 pesan dialokasikan kepada 2.000 *client*, penyesuaian pembulatan akan menghasilkan total pesan sebanyak 16.000. Perbedaan sebanyak 1.000 ini merupakan konsekuensi langsung dari proses pembulatan tersebut.

C. Pengujian Jumlah *Client*

Pengujian ini dirancang bertujuan untuk mengevaluasi efektivitas yang diuji berdasarkan peningkatan jumlah *client* terhadap penggunaan CPU pada protokol HTTP dan MQTT. Parameter yang digunakan dalam pengujian ini dijelaskan secara rinci pada Tabel 3.

Tabel 3. Parameter Pengujian Jumlah *Client*

Properti	Nilai
Pesan Tetap	15.000 Pesan
Penambahan *Client*	500 *Client*

Pesan tetap pada penelitian ini diberikan sebanyak 15.000 pesan digunakan untuk pengujian pada kedua protokol, pengujian dilakukan dengan meningkatkan jumlah *client*, dimulai dari 100, 500, hingga 3.000, dengan kenaikan sebanyak 500 *client* per pengujian. Setiap pengujian untuk setiap jumlah *client* akan dilakukan sebanyak sepuluh kali, kemudian hasil pengujian ditampilkan berupa nilai rata-rata pengujian dalam bentuk tabel dan grafik.

D. Pengujian Jumlah Pesan

Pengujian ini dilakukan bertujuan untuk mengevaluasi dampak peningkatan jumlah pesan terhadap penggunaan CPU pada protokol HTTP dan MQTT, parameter untuk pengujian ini dijelaskan pada Tabel 4.

Tabel 4. Parameter Pengujian Jumlah Pesan

Properti	Nilai
Client Tetap	100 *Client*
Penambahan *Client*	5.000 Pesan 10.000 Pesan 50.000 Pesan 100.000 Pesan 200.000 Pesan 300.000 Pesan 400.000 Pesan 500.000 Pesan 550.000 Pesan 600.000 Pesan

Kedua protokol menggunakan jumlah *client* tetap sebanyak 100 *client*. Jumlah pesan yang diujikan ditingkatkan secara bertahap, dimulai dari 5.000, 10.000, 50.000, 100.000, hingga 600.000, dengan kenaikan sebanyak 100.000. Pengujian akan dihentikan apabila penggunaan CPU mencapai 100% atau mendekati, proses pemantauan aktivitas penggunaan CPU dilakukan dengan menggunakan aplikasi HTOP di Ubuntu 22.04 LTS. Setiap pengujian untuk setiap jumlah pesan akan dilakukan sebanyak sepuluh kali, kemudian hasil pengujian ditampilkan berupa nilai rata-rata pengujian dalam bentuk tabel dan grafik.

E. Pengujian Waktu Pengiriman Pesan

Pengujian ini bertujuan untuk mengukur waktu yang dibutuhkan dalam mentransmisikan seluruh pesan pada server menggunakan protokol HTTP dan MQTT serta membandingkan efisiensinya dalam proses pengiriman data. Parameter yang digunakan dijelaskan secara rinci seperti pada Tabel 4 yang sudah dijelaskan sebelumnya seperti jumlah

pesan, jumlah *client*, serta kondisi jaringan yang stabil selama pengujian. Setiap pengujian untuk menghitung waktu pengiriman pesan akan dilakukan sebanyak sepuluh kali, kemudian hasil pengujian ditampilkan berupa nilai rata-rata pengujian dalam bentuk tabel dan grafik.

F. Persentase Tingkat Penerimaan dan Kehilangan Pesan

Pengujian ini dilakukan untuk menganalisa jumlah pesan yang berhasil diterima pada protokol HTTP dan MQTT. Parameter pengujian ini juga menggunakan skenario pada Tabel 4. Studi terkait sebelumnya telah meneliti persentase tingkat kehilangan pesan (Guha Roy et al., 2018) yang menunjukkan bahwa tingkat kehilangan sekitar 2% masih dianggap dapat diterima atau kondisi normal. Setiap pengujian untuk menghitung tingkat penerimaan dan kehilangan jumlah pesan akan dilakukan sebanyak sepuluh kali, kemudian hasil pengujian ditampilkan berupa nilai rata-rata pengujian dalam bentuk tabel dan grafik.

HASIL DAN PEMBAHASAN

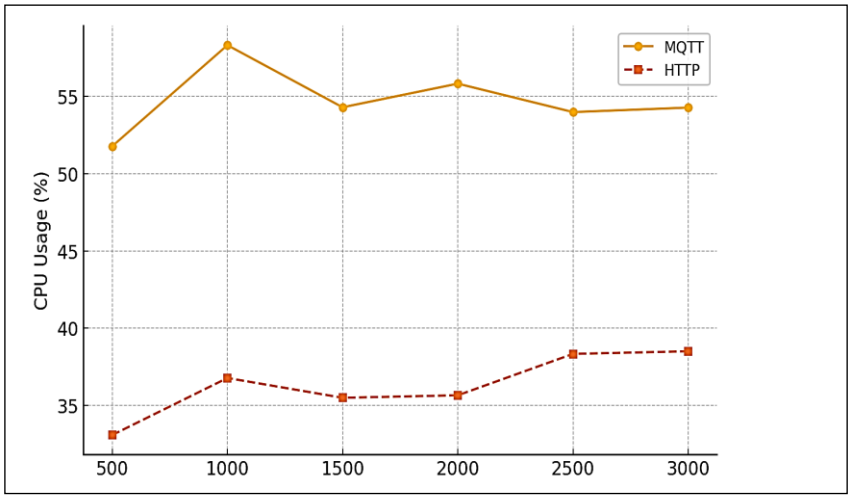
A. Hasil Pengujian Jumlah *Client*

Hasil perbandingan penggunaan CPU antara protokol HTTP dan MQTT seiring dengan peningkatan jumlah *client* disajikan pada Tabel 5.

Tabel 5. Hasil Pengujian Jumlah *Client* Terhadap Penggunaan CPU

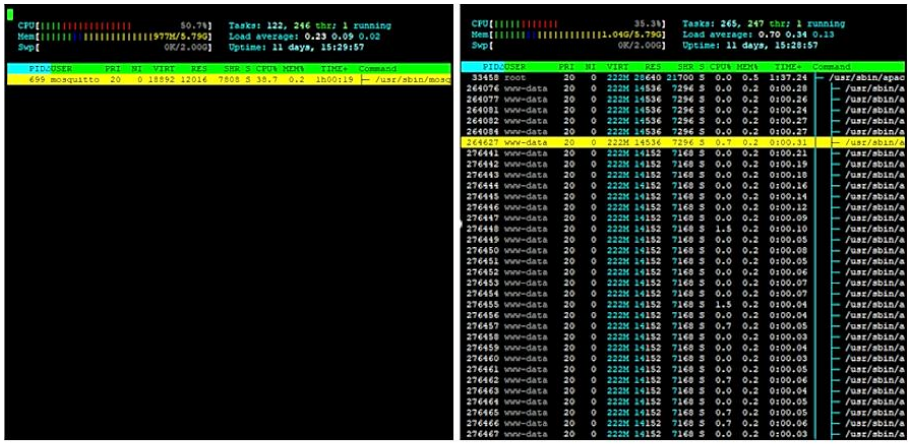
Client	Pesan	Rata-Rata Penggunaan CPU (MQTT)	Rata-Rata Penggunaan CPU (HTTP)
500	15.000	51,78%	33,09%
1.000	15.000	58,32%	36,78%
1.500	15.000	54,3%	35,5%
2.000	15.000	55,83%	35,66%
2.500	15.000	53,99%	38,34%
3.000	15.000	54,28%	38,51%

Pada Tabel 5 merupakan pengujian yang dilakukan dengan meningkatkan jumlah *client* yang dimulai dari 500 hingga 300 dan pengaruhnya pada penggunaan CPU pada masing-masing server. Berdasarkan hasil perhitungan jumlah *client*, terlihat bahwa penggunaan CPU pada protokol MQTT cenderung lebih tinggi dibandingkan dengan HTTP dalam menangani jumlah pesan yang sama, yaitu 15.000 pesan. Ketika jumlah *client* masih relatif rendah, seperti pada 500 *client*, penggunaan CPU untuk MQTT mencapai 51,78%, sedangkan HTTP hanya 33,09%. Seiring bertambahnya jumlah *client* menjadi 1.000, penggunaan CPU MQTT meningkat menjadi 58,32%, sementara HTTP juga mengalami kenaikan menjadi 36,78%. Namun, yang menarik adalah saat jumlah *client* bertambah menjadi 1.500 dan 2.000, penggunaan CPU pada MQTT justru mengalami sedikit penurunan dibandingkan sebelumnya, yaitu menjadi 54,3% dan 55,83%. Sementara itu, HTTP tetap menunjukkan pola kenaikan yang lebih stabil, meskipun peningkatannya relatif kecil, yakni 35,5% dan 35,66%. Hal ini mengindikasikan bahwa MQTT tidak selalu mengalami peningkatan konsumsi CPU secara linier terhadap jumlah *client* yang bertambah, sedangkan HTTP menunjukkan tren yang lebih konsisten. Pada jumlah *client* yang lebih besar, yakni 2.500 dan 3.000, penggunaan CPU MQTT kembali menurun sedikit menjadi 53,99% dan 54,28%, sementara HTTP justru mengalami peningkatan lebih signifikan dibandingkan sebelumnya, yaitu 38,34% dan 38,51%. Dari hasil ini, terlihat bahwa meskipun pada jumlah *client* yang lebih kecil perbedaan antara MQTT dan HTTP cukup besar, semakin banyak *client* yang ditangani, maka selisih penggunaan CPU antara kedua protokol ini semakin mengecil. Grafik penggunaan CPU antara protokol MQTT dan HTTP dapat dilihat pada Gambar 2.



Gambar 2. Grafik Perbandingan *Client* Pada MQTT dan HTTP

Jika dilihat dari sisi server dengan menggunakan HTOP pada Ubuntu 22.04 LTS, terdapat perbedaan cara menangani proses antara server MQTT dan HTTP yang dapat dilihat pada Gambar 3.



MQTT HTTP

Gambar 3. Pemantauan Jumlah Penggunaan CPU Dengan HTOP

Gambar 3 merupakan perbedaan antara Mosquitto dan Apache2 dalam menangani pemrosesan data. Mosquitto memproses data menggunakan pendekatan *single-process*, yang artinya banyak *client* dan pesan ditangani dalam satu proses saja. Sedangkan HTTP menanganinya menggunakan mekanisme *multi-process*, sehingga HTTP dapat menangani banyak proses dengan penggunaan CPU yang lebih stabil. Hal ini menjelaskan mengapa hasil pengujian menunjukkan bahwa protokol MQTT berkinerja lebih tinggi daripada HTTP. Dalam kasus ini, kedua protokol dapat menangani 3.000 *client* atau lebih, namun seiring bertambahnya jumlah *client*, jumlah pesan yang dikirimkan akan berkurang.Secara keseluruhan, dapat disimpulkan bahwa MQTT cenderung membutuhkan lebih banyak sumber daya CPU dibandingkan HTTP dalam menangani jumlah *client* yang sama. Namun, MQTT juga menunjukkan efisiensi yang lebih stabil saat jumlah *client* bertambah, sementara HTTP mengalami kenaikan yang lebih linear dan cenderung meningkat tajam pada jumlah *client* yang lebih besar.

B. Hasil Pengujian Jumlah Pesan

Perbandingan kinerja protokol MQTT dan HTTP, dan pengaruhnya terhadap penggunaan CPU yang dilakukan dengan meningkatkan jumlah pesan yang dikirimkan dapat dilihat pada Tabel 6.

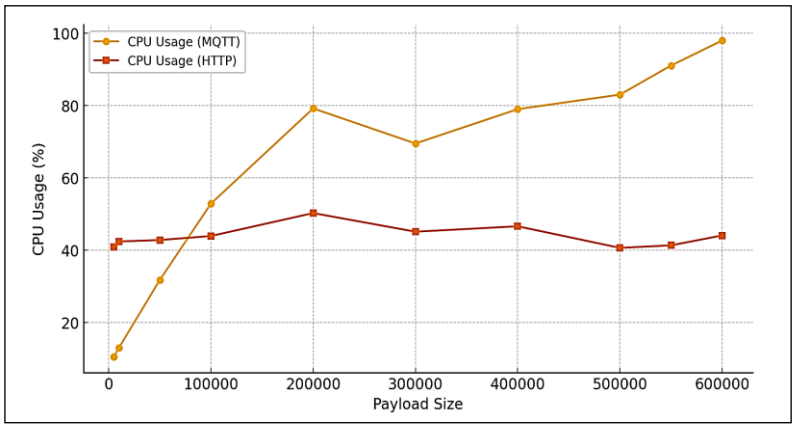
Tabel 6. Hasil Pengujian Jumlah Pesan Terhadap Penggunaan CPU

Client	Pesan	Rata-Rata Penggunaan CPU (MQTT)	Rata-Rata Penggunaan CPU (HTTP)
100	5.000	10,45%	40,82%
100	10.000	12,98%	42,39%
100	50.000	31,75%	42,79%
100	100.000	52,87%	43,91%
100	200.000	79,22%	50,25%
100	300.000	69,51%	45,1%
100	400.000	79,01%	46,6%
100	500.000	83,04%	40,62%
100	550.000	91,09%	41,32%
100	600.000	98,02%	44,06%

Dari hasil perhitungan peningkatan jumlah pesan, terlihat bahwa penggunaan CPU pada protokol MQTT dan HTTP menunjukkan hal yang berbeda seiring bertambahnya jumlah pesan yang dikirim. Pada tahap awal dengan 5.000 pesan, penggunaan CPU untuk MQTT relatif rendah, hanya 10,45%, sedangkan HTTP memiliki penggunaan CPU yang jauh lebih tinggi, yaitu 40,82%. Seiring meningkatnya jumlah pesan menjadi 10.000, penggunaan CPU pada MQTT sedikit meningkat menjadi 12,98%, sementara HTTP juga mengalami kenaikan kecil menjadi 42,39%. Ketika jumlah pesan mencapai 50.000, penggunaan CPU pada server MQTT naik signifikan ke 31,75%, sementara HTTP tetap stabil di 42,79%. Perubahan ini terus berlanjut hingga 100.000 pesan, di mana MQTT mencapai 52,87%, sementara HTTP masih relatif

konstan di 43,91%. Namun, saat jumlah pesan meningkat ke 200.000, penggunaan CPU pada MQTT melonjak drastis menjadi 79,22%, menunjukkan bahwa protokol ini semakin membutuhkan sumber daya CPU pada server seiring bertambahnya jumlah pesan. Sementara itu, HTTP mengalami kenaikan menjadi 50,25%.

Menariknya, ketika jumlah pesan mencapai 300.000, penggunaan CPU pada MQTT justru mengalami penurunan ke 69,51%, sebelum kembali meningkat ke 79,01% pada 400.000 pesan, lalu mencapai 83,04% pada 500.000 pesan. Sementara itu, HTTP menunjukkan fluktuasi yang lebih kecil, dengan penggunaan CPU berkisar antara 40,62% hingga 50,25%, menunjukkan kestabilan yang lebih baik dibandingkan MQTT. Pada jumlah pesan 550.000 dan 600.000, penggunaan CPU MQTT terus meningkat hingga mendekati 98,02%, sedangkan HTTP tetap relatif stabil di sekitar 44,06%. Grafik penggunaan CPU antara protokol MQTT dan HTTP dapat dilihat pada Gambar 4.



Gambar 4. Grafik Perbandingan Jumlah Pesan Pada MQTT dan HTTP

Hal ini juga terjadi karena broker Mosquitto memproses data menggunakan pendekatan single-process, yang artinya banyak *client* dan pesan ditangani dalam satu proses saja. Sedangkan HTTP menanganinya menggunakan mekanisme multi-process, sehingga HTTP dapat menangani banyak proses dengan penggunaan CPU yang lebih stabil. Dari data ini, dapat disimpulkan bahwa MQTT mengalami peningkatan penggunaan CPU yang lebih drastis dibandingkan HTTP seiring bertambahnya jumlah pesan, menunjukkan bahwa meskipun efisien pada jumlah pesan kecil, MQTT cenderung lebih boros sumber daya saat beban pesan meningkat. Sebaliknya, HTTP memiliki pola penggunaan CPU yang lebih stabil, meskipun sejak awal cenderung lebih tinggi dibandingkan MQTT.

C. Hasil Pengujian Waktu Pengiriman Pesan

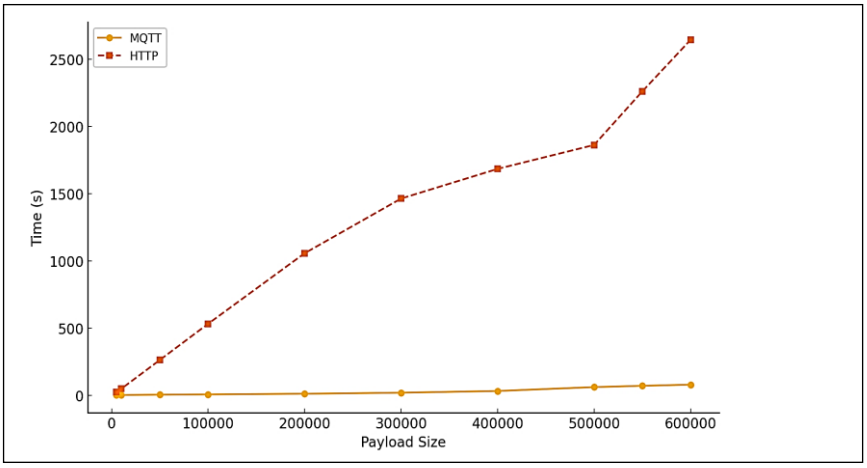
Perbandingan total waktu transfer antara protokol HTTP dan MQTT seiring dengan peningkatan jumlah pesan disajikan pada Tabel 7.

Tabel 7. Hasil Total Waktu Pengiriman

Client	Pesan	Rata-Rata Waktu Pengiriman MQTT (Detik)	Rata-Rata Waktu Pengiriman HTTP (Detik)
100	5.000	2,604	25,447
100	10.000	3,295	49,156
100	50.000	5,313	263,63
100	100.000	7,599	532,528
100	200.000	12,415	1.056,789
100	300.000	20,179	1.463,491
100	400.000	32,876	1.684,001
100	500.000	61,831	1.861,961
100	550.000	71,506	2.260,059
100	600.000	79,927	2.643,359

Berdasarkan hasil pengujian waktu transfer, terlihat bahwa protokol MQTT memiliki kecepatan yang jauh lebih tinggi dibandingkan HTTP dalam mengirimkan jumlah pesan yang sama. Pada tahap awal, dengan 5.000 pesan, waktu transfer menggunakan MQTT hanya 2,604 detik, sedangkan HTTP membutuhkan waktu yang jauh lebih lama, yaitu 25,447 detik. Selisih waktu ini semakin besar seiring bertambahnya jumlah pesan yang dikirim. Ketika jumlah pesan meningkat menjadi 10.000, waktu transfer MQTT sedikit bertambah menjadi 3,295 detik, sementara HTTP membutuhkan hampir 50 detik. Perbedaan ini semakin mencolok pada 50.000 pesan, di mana MQTT hanya membutuhkan 5,313 detik, sedangkan HTTP membutuhkan 263,63 detik, menunjukkan bahwa HTTP mengalami peningkatan waktu transfer yang jauh lebih drastis dibandingkan MQTT.

Saat jumlah pesan mencapai 100.000, MQTT membutuhkan 7,599 detik, sedangkan HTTP memerlukan waktu hingga 532,528 detik. Lonjakan ini terus berlanjut hingga 200.000 pesan, di mana MQTT hanya memerlukan 12,415 detik, sedangkan HTTP meningkat pesat menjadi 1.056,789 detik atau lebih dari 17 menit. Pada 300.000 pesan, perbedaan semakin signifikan, dengan MQTT membutuhkan 20,179 detik, sedangkan HTTP membutuhkan 1.463,491 detik. Ketika jumlah pesan semakin besar, perbedaan waktu transfer antara kedua protokol semakin tajam. Pada 400.000 pesan, MQTT masih cukup cepat dengan waktu 32,876 detik, sementara HTTP meningkat drastis ke 1.684,001 detik. Hal yang sama terjadi pada 500.000 pesan, di mana MQTT membutuhkan 61,831 detik, sedangkan HTTP mencapai 1.861,961 detik. Bahkan pada 550.000 dan 600.000 pesan, MQTT tetap efisien dengan waktu 71,506 detik dan 79,927 detik, sementara HTTP terus mengalami peningkatan drastis hingga 2.260,059 detik dan 2.643,359 detik. Grafik perbandingan waktu transfer antara protokol MQTT dan HTTP dapat dilihat pada Gambar 5.



Gambar 5. Grafik Perbandingan Waktu Pada MQTT dan HTTP

Dari analisis ini, dapat disimpulkan bahwa MQTT jauh lebih efisien dalam hal waktu transfer dibandingkan HTTP, terutama saat jumlah pesan bertambah besar. HTTP mengalami peningkatan waktu transfer yang eksponensial, sedangkan MQTT tetap dalam kisaran yang lebih stabil dan rendah. Oleh karena itu, dalam sistem pemantauan berbasis IoT yang membutuhkan pengiriman data cepat dan efisien, MQTT merupakan pilihan yang lebih unggul dibandingkan HTTP.

D. Persentase Tingkat Penerimaan dan Kehilangan Pesan

Perbandingan kinerja protokol MQTT dan HTTP, dan pengaruhnya terhadap tingkat pesan yang diterima pada server yang dilakukan dengan meningkatkan jumlah pesan yang dikirimkan dapat dilihat pada Tabel 8.

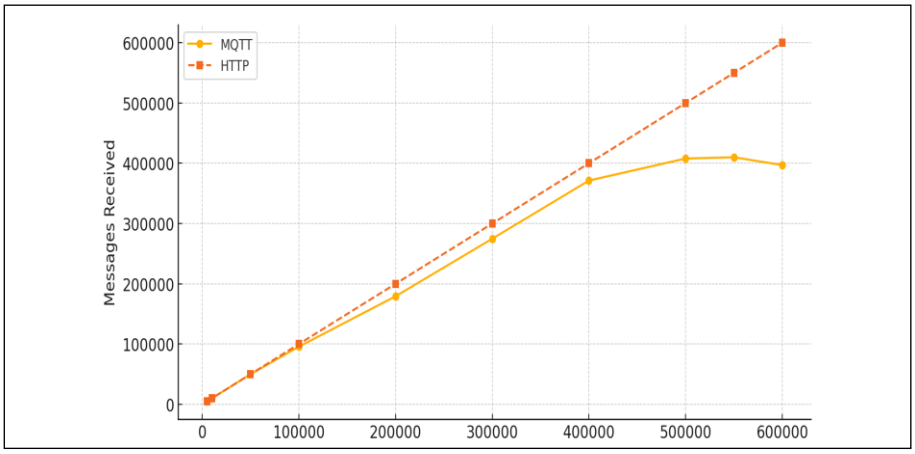
Tabel 8. Hasil Persentase Tingkat Penerimaan dan Kehilangan Pesan

Client	Pesan Dikirim	Pesan Diterima		Persentase Kehilangan Pesan	
		Rata-Rata MQTT	Rata-Rata HTTP	MQTT	HTTP
100	5.000	5.000	5.000	0%	0%
100	10.000	10.000	10.000	0%	0%
100	50.000	50.000	50.000	0%	0%
100	100.000	95.718	100.000	4,282%	0%
100	200.000	179.300	200.000	10,35%	0%
100	300.000	274.797	300.000	8,401%	0%
100	400.000	371.326	400.000	7,1685%	0%
100	500.000	407.913	500.000	18,4174%	0%
100	550.000	410.001	550.000	25,45436%	0%
100	600.000	397.050	600.000	33,825%	0%

Berdasarkan hasil perbandingan tingkat penerimaan pesan, terlihat bahwa protokol HTTP memiliki tingkat keberhasilan penerimaan pesan yang 100% pada semua skenario pengiriman, dari 5.000 pesan hingga 600.000 pesan. Setiap pesan yang dikirim melalui HTTP selalu diterima dengan jumlah yang sama, menunjukkan keandalan protokol ini dalam menjamin transmisi data tanpa kehilangan pesan. Di sisi lain, protokol MQTT menunjukkan performa yang baik pada jumlah pesan kecil, dengan tingkat penerimaan 100% pada 5.000, 10.000, dan 50.000 pesan. Namun, saat jumlah pesan meningkat menjadi 100.000, mulai terjadi kehilangan pesan, dengan hanya 95.718 pesan yang diterima dari 100.000 yang dikirim dan persentase kehilangan data yaitu sebesar 4,282%. Perubahan ini semakin terlihat pada 200.000 pesan, di mana hanya 179.300 pesan yang diterima, dan pada 300.000 pesan, jumlah pesan yang berhasil diterima adalah

274.797.

Saat jumlah pesan terus meningkat, kehilangan pesan pada MQTT semakin jelas. Pada 400.000 pesan, hanya 371.326 pesan yang diterima, sementara pada 500.000 pesan, jumlah pesan yang berhasil diterima hanya 407.913. Bahkan, pada 550.000 pesan, tingkat penerimaan mengalami stagnasi di sekitar 410.001 pesan, dan pada 600.000 pesan, justru menurun menjadi 397.050 pesan dan persentase kehilangan data yaitu sebesar 33,825%. Grafik perbandingan tingkat penerimaan pesan antara protokol MQTT dan HTTP dapat dilihat pada Gambar 6.



Gambar 6. Grafik Perbandingan Tingkat Penerimaan Pesan Pada MQTT dan HTTP

Dari analisis ini, dapat disimpulkan bahwa HTTP lebih andal dalam menjamin penerimaan pesan tanpa kehilangan data, meskipun memiliki waktu transfer yang jauh lebih lama dibandingkan MQTT. Pada protokol MQTT, meskipun lebih cepat dalam pengiriman pesan, mulai mengalami kehilangan data ketika beban pesan meningkat, terutama setelah melewati 100.000 pesan. Oleh karena itu, dalam implementasi sistem berbasis IoT, MQTT lebih cocok digunakan dalam kondisi yang menuntut kecepatan tinggi dengan toleransi kehilangan data tertentu, sementara HTTP lebih sesuai untuk aplikasi yang memerlukan keandalan tinggi tanpa kehilangan pesan.

E. Pembahasan Hasil Perbandingan Kinerja

Berdasarkan hasil pengujian perbandingan kinerja antara protokol MQTT dan HTTP menunjukkan bahwa MQTT cenderung menggunakan lebih banyak CPU dibandingkan HTTP karena sifatnya yang berbasis proses tunggal, sementara HTTP lebih efisien dengan pendekatan multiproses. Hal ini sejalan dengan penelitian yang dilakukan oleh (Hong et al., 2023) yang menyebutkan bahwa protokol MQTT lebih cepat dalam memproses data, namun HTTP menunjukkan kestabilan dalam pemrosesan data. Dalam hal pengelolaan pesan, HTTP lebih stabil saat menangani ukuran muatan yang besar dibandingkan dengan MQTT. Penggunaan CPU pada MQTT meningkat drastis hingga 98,02% seiring bertambahnya ukuran muatan, sedangkan HTTP tetap berada dalam rentang 40%-50%. Dari segi waktu transmisi, MQTT memiliki keunggulan yang signifikan karena mampu mengirim dan menerima data lebih cepat dibandingkan HTTP. Hal ini sesuai dengan definisi MQTT yang menyebutkan bahwa protokol ini merupakan protokol yang cepat dan ringan (Bender et al., 2021), penelitian yang dilakukan oleh (Nugraha et al., 2024) juga menyatakan hal serupa bahwa dalam hal waktu transfer data, MQTT sangat cepat dibandingkan dengan HTTP. Namun, dalam hal tingkat penerimaan dan tingkat kehilangan data, HTTP lebih unggul dengan tingkat keberhasilan 100%, sedangkan MQTT mengalami kehilangan data sebesar 33,825% dari total 600.000 muatan yang dikirim. Hal ini juga sejalan dengan penelitian yang dilakukan oleh (Nugraha et al., 2024) bahwa walaupun protokol MQTT cepat dan ringan, namun protokol MQTT sering terjadi kehilangan data ketika dalam pemrosesan data, khususnya dalam menangani data yang besar. Berdasarkan hasil pengujian, dapat disimpulkan bahwa meskipun MQTT jauh lebih cepat dalam transmisi data, protokol ini kurang optimal dalam menangani jumlah data yang besar secara bersamaan, yang menyebabkan ketidakstabilan. Sebaliknya, HTTP lebih andal dalam menangani beban data besar berkat mekanisme multiprosesnya, meskipun membutuhkan waktu pemrosesan yang lebih lama untuk menerima data dalam skala besar.

Berdasarkan hasil pengujian yang telah dilakukan, apabila ingin mengembangkan sistem IoT yang bersifat real-time dengan mengandalkan kecepatan pengiriman data dan mengabaikan resiko kehilangan data pada saat pemrosesannya maka disarankan dapat menggunakan protokol MQTT sebagai protokol utama. Sebaliknya apabila ingin mengembangkan sistem IoT dengan mengandalkan ketahanan, kestabilan, dan memastikan semua data yang dikirimkan dapat diterima pada server tanpa ada resiko kehilangan paket, maka disarankan penggunaan protokol HTTP sebagai protokol utama.

KESIMPULAN

Perbandingan kinerja antara protokol MQTT dan HTTP menunjukkan bahwa MQTT memiliki penggunaan CPU yang lebih tinggi daripada HTTP karena mekanisme proses tunggalnya (single-proccess), sedangkan HTTP lebih efisien dengan mekanisme multiproses (multi-proccess). Dalam hal penanganan muatan, HTTP lebih stabil saat memproses muatan besar dibandingkan dengan MQTT. MQTT mengalami peningkatan penggunaan CPU hingga 98,02% saat ukuran muatan meningkat, sementara HTTP tetap stabil dalam kisaran 40%-50%. Dalam hal waktu transmisi, MQTT mengungguli HTTP secara signifikan, terbukti jauh lebih cepat dalam transmisi dan penerimaan data. Mengenai tingkat penerimaan, HTTP berhasil menerima 100% dari data yang dikirim, sedangkan MQTT mengalami kehilangan data sebesar 33,825% dari

600.000 muatan. Berdasarkan hasil pengujian, dapat disimpulkan bahwa MQTT adalah protokol yang jauh lebih cepat dalam hal waktu transmisi dibandingkan dengan HTTP. Namun MQTT tetap dapat menangani sejumlah besar data secara bersamaan namun akan berujung pada ketidakstabilan penggunaan sumber daya. Di sisi lain, HTTP lebih andal dan mampu menangani beban data besar karena mekanisme multi-prosesnya, namun memerlukan lebih banyak waktu pemrosesan untuk penerimaan data berskala besar.

REFERENSI

Alshammari, H. H. (2023). The internet of things healthcare monitoring system based on MQTT protocol. *Alexandria Engineering Journal*, *69*, 275–287. https://doi.org/10.1016/j.aej.2023.01.065

Arshad, Q.-A., Khan, W. Z., Azam, F., Khan, M. K., Yu, H., & Zikria, Y. Bin. (2023). Blockchain-based decentralized trust management in IoT: systems, requirements and challenges. *Complex & Intelligent Systems*, *9*(6), 6155–6176. https://doi.org/10.1007/s40747-023-01058-8

Asyhari, M. W., Sigit, R., & Sukaridhoto, S. (2019). Vending Machine Monitoring System Integrated with Webserver. *2019 International Electronics Symposium (IES)*, 556–559.

Bender, M., Kirdan, E., Pahl, M. O., & Carle, G. (2021, January 9). Open-source MQTT evaluation. *2021 IEEE 18th Annual Consumer Communications and Networking Conference, CCNC 2021*. https://doi.org/10.1109/CCNC49032.2021.9369499

Challapalli, S. S. N., Kaushik, P., Suman, S., Shivahare, B. D., Bibhu, V., & Gupta, A. D. (2021). Web Development and performance comparison of Web Development Technologies in Node.js and Python. *2021 International Conference on Technological Advancements and Innovations (ICTAI)*, 303–307. https://doi.org/10.1109/ICTAI53825.2021.9673464

Chen, B., Slyne, F., & Ruffini, M. (2023). Energy Efficient SDN and SDR Joint Adaptation of CPU Utilization Based on Experimental Data Analytics. *ICC 2023 - IEEE International Conference on Communications*.

Guha Roy, D., Mahato, B., De, D., & Buyya, R. (2018). Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) — MQTT-SN protocols. *Future Generation Computer Systems*, *89*, 300–316. https://doi.org/10.1016/j.future.2018.06.040

Hong, S., Kang, J., & Kwon, S. (2023). Performance Comparison of HTTP, HTTPS, and MQTT for IoT Applications. *International Journal of Advanced Smart Convergence*, *12*(1), 9–17. https://doi.org/10.7236/IJASC.2023.12.1.9

Jara Ochoa, H. J., Peña, R., Ledo Mezquita, Y., Gonzalez, E., & Camacho-Leon, S. (2023). Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations. *Sensors*, *23*(10). https://doi.org/10.3390/s23104896

Kadam, S., & Ghule, S. (2023). Comparative Study of MQTT and HTTP. *International Journal of Innovative Research in Science*, *12*(6), 8593. https://doi.org/10.15680/IJIRSET.2023.1206086

Khalifeh, A., Mazunga, F., Nechibvute, A., & Nyambo, B. M. (2022). Microcontroller Unit-Based Wireless Sensor Network Nodes: A Review. *Sensors*, *22*(22), 8937. https://doi.org/10.3390/s22228937

Kopetz, H., & Steiner, W. (2022). Internet of Things. In *Real-Time Systems* (pp. 325–341). Springer International Publishing. https://doi.org/10.1007/978-3-031-11992-7_13

Kumar, R., Rani, S., & Awadh, M. Al. (2022). Exploring the Application Sphere of the Internet of Things in Industry 4.0: A Review, Bibliometric and Content Analysis. *Sensors*, *22*(11), 4276. https://doi.org/10.3390/s22114276

Malik, P. K., Sharma, R., Singh, R., Gehlot, A., Satapathy, S. C., Alnumay, W. S., Pelusi, D., Ghosh, U., & Nayak, J. (2021). Industrial Internet of Things and its Applications in Industry 4.0: State of The Art. *Computer Communications*, *166*, 125–139. https://doi.org/10.1016/j.comcom.2020.11.016

Mishra, B. (2018). Performance evaluation of MQTT broker servers. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10963 LNCS*, 599–609. https://doi.org/10.1007/978-3-319-95171-3_47

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

Nikolov, N. (2020, September 16). Research of MQTT, CoAP, HTTP and XMPP IoT Communication protocols for Embedded Systems. *2020 29th International Scientific Conference Electronics, ET 2020 - Proceedings*. https://doi.org/10.1109/ET50336.2020.9238208

Nugraha, I. R., Putra, W. H. N., & Setiawan, E. (2024). A Comparative Study of HTTP and MQTT for IoT Applications in Hydroponics. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *8*(1), 119–126.

Oliveira, F., Costa, D. G., Assis, F., & Silva, I. (2024). Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning. *Internet of Things*, *26*, 101153. https://doi.org/10.1016/j.iot.2024.101153

Silva, D., Carvalho, L. I., Soares, J., & Sofia, R. C. (2021). A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA. *Applied Sciences*, *11*(11), 4879. https://doi.org/10.3390/app11114879

Yunana, K., Alfa, A. A., Misra, S., Damasevicius, R., Maskeliunas, R., & Oluranti, J. (2021). *Internet of Things: Applications, Adoptions and Components - A Conceptual Overview* (pp. 494–504). https://doi.org/10.1007/978-3-030-73050-5_50

Zimmerling, M., Mottola, L., & Santini, S. (2021). Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. In *ACM Computing Surveys* (Vol. 53, Issue 6). Association for Computing Machinery. https://doi.org/10.1145/3410159