Penerapan Mosquitto MQTT Bridge pada Sistem IoT dengan Raspberry Pi sebagai Gateway

Muhammad Asep Subandri[1], Fajri Profesio Putra[1], Bayu Pratama Agus Kurniawan[1]
Jurusan Teknik Informatika, Politeknik Negeri Bengkalis
msubandri@polbeng.ac.id[1], fajri@polbeng.ac.id[2], bp11721172@gmail.com[3]

Abstract

The MQTT protocol has become a popular choice in the development of IoT systems due to its efficiency and scalability. In complex IoT architectures, multiple MQTT brokers are often required to manage data from various devices. This research focuses on utilizing the Mosquitto MQTT Bridge as a solution to integrate multiple MQTT brokers into a single IoT system. By using Raspberry Pi as a gateway, this research explores the configuration and implementation of the MQTT Bridge to connect distributed MQTT brokers. The results of this research are expected to contribute to improving the flexibility and scalability of IoT systems, as well as providing practical guidance for developers in designing more complex IoT architectures.

Keywords : IoT, MQTT, Mosquitto MQTT Bridge, Raspberry Pi, MQTT Brokers

1. PENDAHULUAN

Internet of Things (IoT) telah menjadi landasan bagi berbagai industri modern, mulai dari manufaktur, pertanian, hingga sistem rumah pintar. IoT memungkinkan perangkat untuk saling terhubung dan berkomunikasi melalui internet, memungkinkan otomatisasi, monitoring, dan kontrol jarak jauh yang lebih efisien (Xu dkk., 2014; Yang dkk., 2019). Untuk mengelola komunikasi antar perangkat IoT yang sering kali tersebar secara geografis, diperlukan protokol komunikasi yang ringan dan efisien. *Message Queuing Telemetry Transport* (MQTT), yang dikembangkan oleh IBM pada akhir 1990-an, merupakan salah satu protokol komunikasi yang paling populer digunakan dalam IoT. MQTT dirancang untuk memungkinkan komunikasi antar perangkat dengan konsumsi bandwidth yang rendah dan daya yang efisien, menjadikannya pilihan yang ideal untuk perangkat IoT yang memiliki keterbatasan sumber daya (Has dkk., 2023).

Namun, seiring dengan berkembangnya sistem IoT yang semakin kompleks dan tersebar, masalah skalabilitas dan keterbatasan komunikasi antar broker MQTT menjadi tantangan yang signifikan. Sistem IoT yang besar sering kali terdiri dari banyak perangkat dan beberapa broker MQTT untuk mengatur aliran data secara terdistribusi. Dalam situasi ini, broker-broker tersebut tidak dapat berkomunikasi secara langsung tanpa adanya mekanisme tertentu yang menghubungkan mereka. Salah satu solusi yang dapat diterapkan untuk mengatasi masalah ini adalah MQTT Bridge.

MQTT Bridge merupakan mekanisme yang memungkinkan dua atau lebih broker MQTT untuk saling berkomunikasi dengan cara menghubungkan pesan yang dikirimkan dari satu broker ke broker lainnya. Ini memberikan keuntungan dalam hal skalabilitas, di mana sistem IoT yang menggunakan banyak broker dapat diperluas dengan mudah tanpa perlu mengubah arsitektur yang ada (Ruenpitak dkk., 2022). Setiap broker dalam sistem tetap bertanggung jawab atas komunikasi di wilayah atau kelompok perangkatnya sendiri, sementara pesan yang penting dapat diteruskan ke broker lain melalui bridge. Hal ini memungkinkan sistem IoT untuk menangani peningkatan jumlah perangkat secara lebih efisien tanpa membebani satu *broker* tunggal dengan semua lalu lintas komunikasi.

Selain itu, MQTT Bridge juga memberikan fleksibilitas dalam pengelolaan perangkat IoT yang tersebar di berbagai lokasi. Misalnya, dalam aplikasi *smart city* atau *monitoring* industri skala besar, perangkat IoT di setiap zona atau lokasi dapat dikelola oleh broker MQTT yang terpisah. *Bridge* antar broker memungkinkan informasi penting seperti status perangkat, peringatan, atau data sensor untuk dibagikan di seluruh sistem, tanpa mengharuskan perangkat untuk berkomunikasi secara langsung dengan *broker* di lokasi yang jauh. Dengan demikian, sistem yang dibangun menggunakan MQTT Bridge dapat berfungsi dengan lebih terintegrasi dan efektif.

Raspberry Pi, dengan keunggulannya sebagai perangkat komputasi mini yang fleksibel dan murah, merupakan *platform* yang ideal untuk diimplementasikan sebagai *broker* MQTT yang mendukung MQTT Bridge. Raspberry Pi dapat digunakan dalam berbagai aplikasi IoT skala kecil hingga menengah, dan ketika dikonfigurasi sebagai *broker* yang mendukung *bridge*, ia dapat meningkatkan kemampuan komunikasi antar perangkat IoT di sistem terdistribusi.

Dalam sistem yang terdistribusi, keterbatasan jangkauan broker MQTT menjadi hambatan utama dalam mengelola perangkat IoT di beberapa wilayah geografis yang berbeda. Misalnya, dalam aplikasi pertanian cerdas (*smart farming*) atau sistem transportasi cerdas, satu broker mungkin tidak dapat mencakup seluruh perangkat di area yang sangat luas. Dengan menggunakan MQTT Bridge, *broker-broker* di area yang berbeda dapat dihubungkan sehingga komunikasi data antar perangkat di lokasi yang berbeda dapat terjadi dengan lancar tanpa hambatan.

Penelitian ini bertujuan untuk mengimplementasikan dan mendokumentasikan proses konfigurasi MQTT Bridge pada Raspberry Pi sebagai broker MQTT. Fokus utama adalah pada eksperimen teknis penerapan MQTT Bridge, yang akan mencakup instalasi, konfigurasi, dan pengujian komunikasi antar broker di lingkungan IoT yang terdistribusi. Diharapkan hasil dari penelitian ini dapat memberikan panduan teknis yang jelas tentang bagaimana mengoptimalkan komunikasi antar broker dalam sistem IoT melalui penerapan MQTT Bridge pada Raspberry Pi.

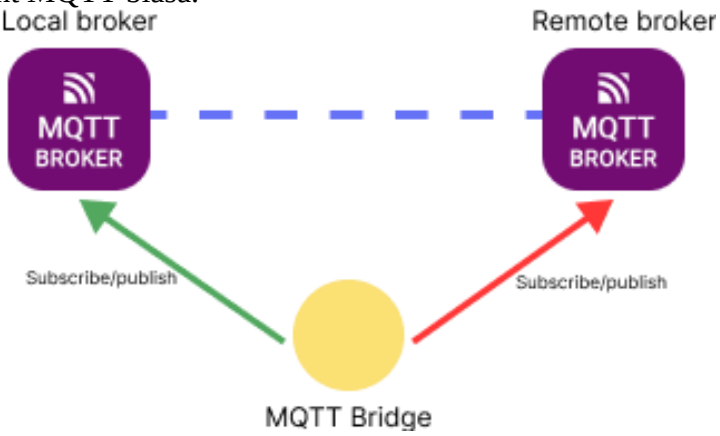
2. TINJAUAN PUSTAKA

Message Queuing Telemetry Transport (MQTT) pertama kali dikembangkan oleh Dr. Andy Stanford-Clark dari IBM dan Arlen Nipper dari Eurotech pada tahun 1999. Protokol ini dirancang khusus untuk komunikasi yang efisien di jaringan dengan *bandwidth* terbatas, latensi tinggi, atau yang tidak dapat diandalkan, seperti jaringan satelit atau seluler (Chen & Kunz, 2016). Seiring berjalannya waktu, MQTT menjadi pilihan utama untuk aplikasi IoT yang membutuhkan komunikasi ringan dan andal. MQTT menggunakan model *komunikasi publish-subscribe*, di mana *client* (*publisher*) dapat mengirim pesan ke topik tertentu, dan *client* lain (*subscriber*) yang berlangganan ke topik tersebut akan menerima pesan yang dipublikasikan. Di antara client ini, terdapat entitas perantara yang disebut *broker* yang mengatur distribusi pesan.

Namun, seiring dengan meningkatnya penggunaan sistem IoT yang berskala besar dan terdistribusi, keterbatasan dari model *single-broker* menjadi semakin jelas. *Single broker* tidak dapat dengan mudah menangani volume lalu lintas data yang besar dari ribuan perangkat IoT yang tersebar di berbagai lokasi geografis tanpa mengalami penurunan kinerja atau masalah jaringan. Selain itu, *single broker* juga memiliki potensi menjadi *single point of failure*, yang berarti jika *broker* mengalami kegagalan, seluruh sistem akan terpengaruh. Dalam konteks inilah muncul kebutuhan untuk meningkatkan skalabilitas sistem IoT dengan membagi tugas *broker* ke beberapa lokasi terpisah yang masih dapat saling berkomunikasi.

Untuk mengatasi tantangan ini, konsep MQTT Bridge diperkenalkan sebagai mekanisme untuk menghubungkan dua atau lebih *broker* dalam sistem yang terdistribusi (Longo dkk., 2022). MQTT Bridge memungkinkan *broker* yang berbeda untuk bertukar pesan antar perangkat IoT, sehingga memperluas cakupan dan kemampuan sistem tanpa harus

mengubah arsitektur yang mendasarinya. Dalam pengaturan ini, *broker* berfungsi sebagai simpul di jaringan yang lebih besar, dan *bridge* bertindak sebagai penghubung untuk memastikan komunikasi antar broker tetap berjalan lancar. Gambar 1 berikut merupakan ilustrasi dari cara kerja MQTT Bridge. Ketika MQTT Broker dikonfigurasikan menjadi *Bridge*, maka ia akan menjadi client dari *remote broker* dan *subcribe*/*publish* ke topik pada broker lainnya seperti client MQTT biasa.



Gambar 1. Cara kerja MQTT Bridge

MQTT Bridge juga memberikan keuntungan dalam hal redundansi dan toleransi kesalahan. Jika satu *broker* mengalami kegagalan, *broker* lain dalam jaringan yang dihubungkan oleh *bridge* dapat terus melayani perangkat yang tersambung tanpa kehilangan data penting (Schmitt dkk., 2018). Penggunaan *bridge* ini juga memungkinkan isolasi lalu lintas data untuk meningkatkan keamanan dan kinerja, karena setiap *broker* dapat mengelola sekumpulan perangkat secara lokal dan hanya meneruskan data penting melalui *bridge* ke *broker* lain. Konfigurasi ini memberikan keleluasaan kepada pengembang IoT untuk merancang sistem yang lebih tangguh dan skalabel.

Raspberry Pi adalah komputer mini yang telah banyak digunakan dalam berbagai proyek IoT, termasuk sebagai *platform* untuk menjalankan broker MQTT. Dengan menggunakan perangkat lunak seperti Mosquitto, Raspberry Pi dapat dikonfigurasi untuk berfungsi sebagai *broker* MQTT yang handal dan hemat biaya. Raspberry Pi sangat cocok untuk tugas-tugas seperti pengelolaan sensor dan perangkat IoT yang tersebar luas. Selain itu, Raspberry Pi menawarkan fleksibilitas untuk digunakan dalam berbagai skenario, termasuk sebagai *platform edge computing*, di mana data dari sensor lokal dapat diproses sebelum dikirim ke broker MQTT (Mahmud & Toosi, 2022).

MQTT Bridge yang diimplementasikan pada Raspberry Pi memberikan solusi hemat biaya untuk mengelola komunikasi dalam sistem IoT yang terdistribusi. Raspberry Pi memungkinkan pengembang untuk membangun sistem *broker* terdistribusi dengan biaya minimal, sambil tetap mempertahankan fleksibilitas dan skalabilitas yang tinggi dalam manajemen perangkat IoT.

3. METODE PENELITIAN

Alur Metodologi

Penelitian ini akan berfokus pada tahapan implementasi teknis MQTT Bridge pada Raspberry Pi, dimulai dari instalasi dan konfigurasi *broker* MQTT hingga penerapan dan pengujian komunikasi antar broker yang saling terhubung. Metode ini terdiri dari beberapa langkah utama, yang meliputi:

1. Instalasi Perangkat Lunak: Menginstal sistem operasi Raspberry Pi OS pada dua perangkat Raspberry Pi. Setelah itu, menginstal *broker* MQTT berbasis Mosquitto pada kedua perangkat.

2. Konfigurasi MQTT Broker: Melakukan konfigurasi masing-masing broker MQTT untuk satu Raspberry Pi sebagai broker utama dan Raspberry Pi lainnya sebagai *broker bridge*. *Broker bridge* akan dikonfigurasi untuk meneruskan pesan antara dua jaringan yang terpisah.
3. Eksperimen Implementasi: Menguji komunikasi antar *broker* dengan menghubungkan perangkat IoT atau simulasi client MQTT yang terhubung ke *broker* masing-masing.
4. Pengumpulan Data Eksperimen: Memverifikasi keberhasilan komunikasi dengan melakukan pengujian *subscribe* dan *publish* antar *broker* melalui *bridge* yang dikonfigurasi.

Instalasi Perangkat Lunak

Pada tahap awal, dua Raspberry Pi akan diatur dengan sistem operasi Raspberry Pi OS (32-bit) yang dapat diunduh dari situs resmi Raspberry Pi. Langkah-langkah instalasi mencakup:

1. Memasang Raspberry Pi OS pada kartu SD menggunakan *tool* Raspberry Pi Imager.
2. Menghubungkan Raspberry Pi ke jaringan lokal melalui Wi-Fi atau *Ethernet*.

Setelah sistem operasi siap, broker MQTT Mosquitto akan diinstal pada kedua Raspberry Pi. Perintah berikut digunakan untuk menginstal Mosquitto:

```
sudo apt update
sudo apt install mosquitto mosquitto-clients
```

Setelah instalasi selesai, Mosquitto akan dikonfigurasi untuk mendukung *bridge* MQTT pada salah satu Raspberry Pi.

Konfigurasi Broker MQTT dengan MQTT Bridge

Untuk mengonfigurasi MQTT Bridge, kita perlu memodifikasi file konfigurasi mosquitto.conf pada Raspberry Pi yang akan berfungsi sebagai bridge. File mosquitto.conf ini bertindak sebagai pusat pengaturan untuk broker Mosquitto, di mana pengaturan seperti *listener, log,* dan *bridge* dapat ditetapkan. Berikut adalah format umum dari file mosquitto.conf:

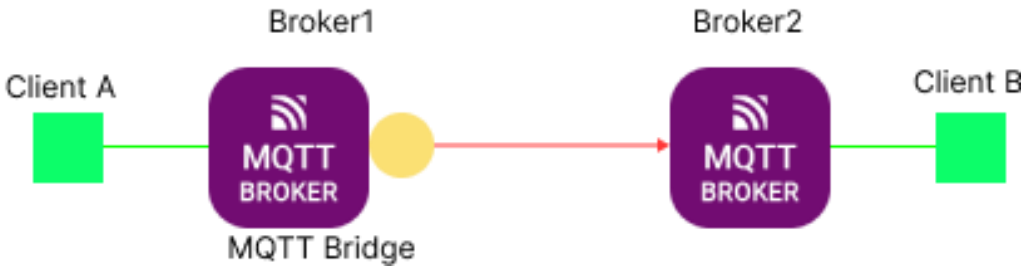
```
listener 1883

allow_anonymous true

log_dest file /var/log/mosquitto/mosquitto.log
log_type all

connection <bridge_name>
address <broker2_ip>:1883
topic <topic> in|out|both
cleansession true
try_private true
start_type automatic
```

Penjelasan dari konfigurasi di atas:

a. *listener*: Menentukan port yang akan digunakan broker (misalnya, port 1883 untuk MQTT).
b. *allow_anonymous*: Jika diset ke *true*, memungkinkan *client* untuk terhubung tanpa otentikasi.
c. *log_dest* dan *log_type*: Menentukan log output untuk *broker*, berguna untuk debugging.
d. *connection*: Menandai awal dari pengaturan *bridge*, diikuti oleh nama *bridge*.
e. *address*: Alamat IP dan port broker MQTT tujuan.
f. *topic*: Menentukan topik yang akan dijembatani antara broker. Opsi in, out, atau both menentukan arah pesan (*inbound, outbound*, atau keduanya).

g. *cleansession*: Jika diset ke true, client tidak akan menyimpan sesi sebelumnya.
h. *try_private*: Mengatur apakah bridge menggunakan sesi privat atau tidak.
i. *start_type*: Menentukan apakah bridge harus diaktifkan secara otomatis ketika broker dimulai.

Misalkan kita memiliki dua *broker*, broker1 (di Raspberry Pi 1) dan broker2 (di Raspberry Pi 2) seperti yang ditunjukkan oleh Gambar 2 di bawah ini.



Gambar 2. Konfigurasi MQTT Bridge untuk eksperimen

Broker1 akan diatur sebagai *bridge*, yang mengirim dan menerima pesan dari broker2. Berikut adalah contoh konfigurasi mosquitto.conf untuk broker1 (Raspberry Pi 1):

```
# Konfigurasi broker1 (Raspberry Pi 1)
listener 1883
allow_anonymous true
log_dest file /var/log/mosquitto/mosquitto.log
log_type all

# Konfigurasi bridge
connection broker2_bridge
address 192.168.1.20:1883   # Alamat IP broker2 (Raspberry Pi 2)
topic sensor_data both    # Menghubungkan topik 'sensor_data' di kedua arah
cleansession true
try_private true
start_type automatic
```

Pada broker2, tidak ada pengaturan tambahan untuk *bridge* yang diperlukan. *Broker* ini akan bertindak sebagai broker MQTT biasa yang menerima pesan dari broker1.

Eksperimen Implementasi Dua Broker MQTT pada Dua Raspberry Pi

Setelah konfigurasi selesai, eksperimen dilakukan untuk menguji komunikasi antar broker menggunakan skenario berikut:

1. Inisialisasi *broker*: Jalankan kedua *broker* di Raspberry Pi 1 dan Raspberry Pi 2 dengan perintah berikut:
   ```
   sudo systemctl start mosquitto
   ```
2. Koneksi perangkat IoT atau client MQTT: Hubungkan client atau perangkat IoT ke masing-masing broker. *Client A* terhubung ke broker1 dan *client B* terhubung ke broker2.
3. Pengujian komunikasi: Lakukan pengujian publish dan subscribe menggunakan mosquitto_pub dan mosquitto_sub:
   a. *Client A* yang terhubung ke broker1 melakukan publish ke topik sensor_data:
      ```
      mosquitto_pub -h localhost -t sensor_data -m "Test message from broker1"
      ```
   b. *Client B* yang terhubung ke broker2 akan menerima pesan tersebut melalui bridge:
      ```
      mosquitto_sub -h localhost -t sensor_data
      ```

4. Verifikasi log: Periksa log Mosquitto di kedua *broker* untuk memastikan pesan telah berhasil dikirimkan dan diteruskan antar *broker*.
5. Pengulangan pengujian: Langkah-langkah di atas diulangi dengan berbagai topik dan arah pesan (misalnya, *client B* melakukan publish di broker2 dan *client A* menerima di broker1).

4. HASIL PENELITIAN DAN PEMBAHASAN

Eksperimen melibatkan dua Raspberry Pi, masing-masing dikonfigurasi sebagai *broker* MQTT. Raspberry Pi pertama (broker1) diatur sebagai *bridge* yang berfungsi meneruskan pesan ke Raspberry Pi kedua (broker2), yang berfungsi sebagai *broker* MQTT biasa. Selama eksperimen, berbagai pengujian dilakukan untuk memverifikasi komunikasi antara kedua *broker* dengan menggunakan topik yang sama (sensor_data). Koneksi antar *broker* melalui *bridge* MQTT berhasil dilakukan. Kedua *broker* berhasil berkomunikasi menggunakan topik sensor_data, di mana pesan yang dipublish oleh *client* yang terhubung ke *broker1* dapat diterima oleh client yang terhubung ke *broker2*, dan sebaliknya. Tidak ada pesan yang hilang dalam komunikasi ini.

Log broker pada kedua Raspberry Pi menunjukkan komunikasi yang berhasil dan proses bridging yang terjadi antara kedua broker. Pesan yang diterima oleh broker1 sebagai *bridge* diteruskan ke broker2 sesuai dengan pengaturan direction dalam file konfigurasi mosquitto.conf. Berikut contoh log dari broker1 saat meneruskan pesan ke broker2:

```
1623421524: Received PUBLISH from mosqpub|clientA (d0, q0, r0, m0, 'sensor_data', ... (12 bytes))
1623421524: Sending PUBLISH to broker2_bridge (d0, q0, r0, m0, 'sensor_data', ... (12 bytes))
```

Broker2 juga mencatat penerimaan pesan dari broker1:

```
1623421524: Received PUBLISH from broker1_bridge (d0, q0, r0, m0, 'sensor_data', ... (12 bytes))
1623421524: Sending PUBLISH to mosqsub|clientB (d0, q0, r0, m0, 'sensor_data', ... (12 bytes))
```

Salah satu keuntungan utama dari penerapan MQTT Bridge adalah skalabilitas sistem komunikasi. Dengan menggunakan bridge, sistem dapat dipecah menjadi beberapa *broker* yang terpisah secara geografis atau jaringan, namun tetap dapat berkomunikasi melalui topik-topik yang dijembatani. Hal ini membuka peluang untuk membangun jaringan IoT yang lebih luas dengan distribusi beban yang lebih merata di antara *broker*.

Pada eksperimen ini, satu topik (sensor_data) dijembatani antara dua *broker*. Dalam implementasi nyata, sistem dapat diperluas dengan menggunakan lebih banyak topik dan *broker* tambahan, sehingga sistem lebih fleksibel dalam mengakomodasi peningkatan jumlah perangkat IoT yang terhubung.

Konfigurasi MQTT Bridge menyediakan fleksibilitas yang tinggi dalam hal pengaturan topik dan arah komunikasi. Pada eksperimen ini, arah komunikasi diatur sebagai *both*, memungkinkan pesan dikirimkan dan diterima dari kedua *broker* secara dua arah. Dalam skenario tertentu, pengaturan in atau out dapat digunakan untuk mengontrol lalu lintas pesan, misalnya hanya mengizinkan pesan dari broker bridge diteruskan ke broker lainnya tanpa memungkinkan pesan dari broker tersebut kembali.

Meskipun MQTT Bridge menawarkan banyak manfaat dalam hal fleksibilitas dan skalabilitas, beberapa kendala potensial yang mungkin muncul dalam implementasi jangka panjang adalah sebagai berikut:

a. Keterbatasan sumber daya pada perangkat seperti Raspberry Pi: Karena Raspberry Pi memiliki keterbatasan sumber daya komputasi, *broker* yang ditempatkan pada Raspberry Pi mungkin mengalami penurunan kinerja ketika menangani lalu lintas pesan yang sangat besar.

b. Kompleksitas konfigurasi: Penambahan *broker* dan *bridge* tambahan dapat meningkatkan kompleksitas konfigurasi, terutama jika topik yang dijembatani semakin banyak dan bervariasi.
c. Keamanan: Karena bridge memungkinkan komunikasi antara broker, aspek keamanan seperti otentikasi dan enkripsi data perlu diperhatikan untuk mencegah akses yang tidak diinginkan.

5. KESIMPULAN DAN SARAN

Penerapan MQTT Bridge pada Raspberry Pi telah terbukti berhasil dalam menghubungkan dua broker MQTT, memungkinkan komunikasi yang efisien antara perangkat yang terhubung. Hasil eksperimen menunjukkan bahwa kedua *broker* dapat saling bertukar pesan melalui topik yang sama tanpa kehilangan data, serta menjamin fleksibilitas dan skalabilitas sistem komunikasi. Penggunaan MQTT Bridge juga memungkinkan pengaturan arah komunikasi yang berbeda, memberikan keleluasaan dalam merancang arsitektur jaringan IoT yang lebih kompleks. Meskipun eksperimen ini menunjukkan hasil yang positif, tantangan seperti keterbatasan sumber daya pada Raspberry Pi dan kompleksitas konfigurasi perlu diperhatikan dalam implementasi skala besar.

Untuk penelitian kedepannya, disarankan untuk meningkatkan sumber daya perangkat yang digunakan, seperti mempertimbangkan penggunaan perangkat dengan kapasitas lebih besar atau membentuk cluster Raspberry Pi, guna mengatasi keterbatasan yang mungkin timbul saat menangani lalu lintas pesan yang tinggi. Selain itu, penting untuk menerapkan mekanisme keamanan seperti TLS/SSL untuk mengenkripsi komunikasi antara *broker*, sehingga data yang dipertukarkan tetap aman. Pengembangan arsitektur jaringan juga perlu dieksplorasi lebih lanjut dengan menambah jumlah broker dan menggunakan variasi topik untuk meningkatkan fleksibilitas sistem dalam mengakomodasi lebih banyak perangkat. Selanjutnya, sebagai rencana penelitian ke depan, fokuskan pada pengembangan sistem remapping pada MQTT Bridge, yang memungkinkan pengubahan topik pesan secara dinamis selama proses pengiriman. Langkah ini akan meningkatkan adaptabilitas sistem komunikasi IoT terhadap perubahan kebutuhan yang terjadi.

6. DAFTAR PUSTAKA

Chen, Y., & Kunz, T. (2016). Performance evaluation of IoT protocols under a constrained wireless access network. *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, 1–7. https://doi.org/10.1109/MoWNet.2016.7496622

Has, M., Kušek, M., & Žarko, I. P. (2023). Evaluating Secure Variants of the MQTT Protocol on Resource-Constraint Devices for Precision Agriculture. *2023 17th International Conference on Telecommunications (ConTEL)*, 1–7. https://doi.org/10.1109/ConTEL58387.2023.10199102

Longo, E., Redondi, A. E. C., Cesana, M., & Manzoni, P. (2022). BORDER: A Benchmarking Framework for Distributed MQTT Brokers. *IEEE Internet of Things Journal*, *9*(18), 17728–17740. https://doi.org/10.1109/JIOT.2022.3155872

Mahmud, R., & Toosi, A. N. (2022). Con-Pi: A Distributed Container-Based Edge and Fog Computing Framework. *IEEE Internet of Things Journal*, *9*(6), 4125–4138. https://doi.org/10.1109/JIOT.2021.3103053

Ruenpitak, C., Phonphoem, A., Jansang, A., Tangtrongpairoj, W., & Jaikaeo, C. (2022). Scalable Distributed Broker System for Very Large MQTT Networks. *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 1–6. https://doi.org/10.1109/JCSSE54890.2022.9836246

Schmitt, A., Carlier, F., & Renault, V. (2018). Dynamic bridge generation for IoT data exchange via the MQTT protocol. *Procedia Computer Science*, *130*, 90–97. https://doi.org/10.1016/j.procs.2018.04.016

Xu, L. Da, He, W., & Li, S. (2014). Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, *10*(4), 2233–2243. https://doi.org/10.1109/TII.2014.2300753

Yang, H., Kumara, S., Bukkapatnam, S. T. S., & Tsung, F. (2019). The internet of things for smart manufacturing: A review. *IISE Transactions*, *51*(11), 1190–1216. https://doi.org/10.1080/24725854.2018.1555383