ISSN: 2614-8404

Optimasi Jalur Transfer Data dari HTTP menjadi MQTT pada IoT menggunakan *Cloud Services*

Claudia Felicia Permatasari [1], Harry Dhika [2]
[1] Fakultas Teknik Ilmu Komputer, Prodi Informatika, Universitas Indraprasta PGRI
[2] Fakultas Teknik Ilmu Komputer, Prodi Informatika, Universitas Indraprasta PGRI
Email: claudia.ak47@gmail.com, dhikatr@yahoo.com

Abstrak **Saat ini, di era perkembangan teknologi dan penyebaran informasi yang kian pesat, diperkenalkan sistem perangkat pintar berbasis internet yang dapat membantu meringankan pekerjaan manusia, teknologi ini dikenal dengan sebutan *IoT* atau *Internet of Things*. Dengan adanya teknologi *IoT*, setiap perangkat yang terkoneksi dengan *microcontroller*, bisa kita kendalikan untuk melakukan suatu pekerjaan ataupun *monitoring* dengan mudah dimana saja, kapan saja, dan dengan macam-macam variasi *gadget* yang kita punya. Namun, hal tersebut harus didukung dengan adanya akses internet penunjang yang stabil. Protokol *HTTP* yang mulanya diterapkan untuk menangani jaringan komunikasi dan lalu lintas data antar perangkat pada *IoT* ternyata memiliki berbagai kendala terkait efektivititas dan sumber daya, diantaranya kendala dalam hal pengiriman yang data lebih lambat, sumber daya yang digunakan lebih besar, format pesan yang rumit, ukuran data yang besar, serta proses pengecekan pada lalu lintas tiap data. Hal ini tentu sangat mempengaruhi kinerja sistem terkait. Mengacu pada kendala tersebut, pada jurnal ini akan dibahas komparasi antara protokol *HTTP* dengan *MQTT* dari segi model protokol, kecepatan pengiriman data, ukuran pesan yang dikirimkan, serta penggunaan optimal untuk masing-masing protokol. Implementasi pengujian sistem ini juga melibatkan penggunaan *RabbitMQ* selaku *message broker* antara pengirim/ *sender*/ *publisher* (ponsel pintar) dan penerima/ *receiver*/ *subscriber* (perangkat *IoT*) yang mana bertugas menampung data sementara sebelum didistribusikan kepada *receiver*. Tidak hanya itu, *RabbitMQ* juga berperan menangani apabila koneksi internet tidak stabil ataupun mati untuk sementara waktu, maka *server* dapat menyimpan data sementara pada *RabbitMQ* dan akan kembali meneruskan data tersebut kepada perangkat *IoT* apabila koneksi internet telah stabil. Mekanisme sistem seperti ini tentunya dapat menjadi solusi bagi kita terhadap permasalahan kehilangan data akibat koneksi internet yang tidak stabil. Pengguna tidak lagi takut data akan hilang karna semua data akan tertampung pada *RabbitMQ*, tidak dipengaruhi oleh kestabilan koneksi jaringan internet. Dalam hal ini, *RabbitMQ* harus berada dalam lingkungan *container*, untuk itu dibutuhkan juga sistem *docker* pada saat menginstall *broker* yang mana berguna sebagai media *containerize*. Dengan adanya *docker container*, penginstallan *RabbitMQ* menjadi sedemikian mudah karna penginstallan tidak benar-benar dilakukan dari awal (yang mana kita ketahui berpotensi cukup rumit), melainkan kita hanya akan melakukan penginstallan *container* yang telah terisi oleh *RabbitMQ*.**
Kata Kunci **– Internet of Things, MQTT protocol, RabbitMQ, Cloud Server, HTTP protocol, Docker Container.**

Abstract – At this time, in the era of technological development and information that incresingly very fast, introduced smart device systems that can help any stuff and conditions of human work, this technology is known as IoT or the abbreviation of Internet of Things Technology. With the existence of IoT systems, every devices connected to microcontroller, and we can communicate to do monitoring stuff and work more easier anytime, anywhere, and with variety of gadgets that we have. However, it must be supported by the existence of a stable supporting internet access. The HTTP protocol is used to connect communication networks and data traffic between devices on IoT has laxity with effectiveness, supports faster data transmission capabilities, resources intensive used, complicated message formats, large data sizes, and the process of checking traffic for each data consume a lot of time. This exactly affects for the performance of related systems. Referring to these constraints, this paper will discuss the comparison between HTTP and MQTT protocols in terms of protocol models, speed of data transmission, size of messages sent, and optimal use for each protocol. Implementation of this system test also involves the implementation of RabbitMQ as a Message Broker between the sender / publisher (smart phone) and the receiver / subscriber (IoT device) which is tasked with storing temporary data before being distributed to the receiver. Not only that, RabbitMQ also has a role to handle if the internet connection is unstable or temporarily dead, the server can store data temporarily on RabbitMQ and will return the data to the IoT device if the internet connection is more stable. This such a system mechanism can certainly be a solution for us to against the problem of data loss due to unstable internet connections. Users/ clients are no longer afraid that data will be lost because all data will be accommodated in RabbitMQ stored place, not influenced by the stability of the internet network connection. In this case, RabbitMQ must be in a container environment, for that we also need a docker when installing a broker which is useful as a media containerize. With the docker container, installing RabbitMQ becomes so easy because the installation is not really done from the beginning (as we knew that potentially quite complicated), but we will only install containers that have been filled by RabbitMQ.
Keywords – Internet of Things, MQTT protocol, RabbitMQ, Cloud Server, HTTP protocol, Docker Container.

I. PENDAHULUAN

Di era saat ini, pemanfaatan teknologi perangkat pintar dijadikan sebagai ide cemerlang bagi para inovator dalam melakukan pemecahan masalah. Hal ini dibuktikan dengan adanya komunikasi *machine to machine* (M2M) yang terjadi antara dua perangkat atau lebih melalui jaringan internet, yang dimanfaatkan untuk memudahkan pekerjaan manusia. Perangkat-perangkat ini nantinya saling terhubung dan dapat berkomunikasi satu sama lainnya melalui jaringan internet secara global menggunakan protokol dan skema / pola komunikasi tertentu tergantung proses data yang mendasari tiap aplikasi. Teknologi *IoT* ini tentunya memiliki keterbatasan fungsi dan jangkauan, terbukti dari batas penyimpanan (*storage*) yang dimiliki, serta kemampuan sistem dalam melakukan komputasi. Namun kendala-kendala tersebut tidak lantas membuat

ISSN: 2614-8404

pengembang meninggalkan teknologi ini begitu saja, ada banyak sistem penunjang yang dapat menambal kelemahan dari teknologi *IoT* ini. Contohnya antara lain teknologi *cloud computing* yang dapat menunjang kemampuan komputasi sistem lebih baik, *RabbitMQ* sebagai salah satu *message broker* yang dapat membantu mengatur manajemen *flow data*, serta *docker container* yang memudahkan pengembang sistem dalam membangun *environment* sistem. Hal ini mejadikan teknologi *IoT* lebih berkembang pesat saat ini dan untuk dikemudian hari dikarenakan pola pengembangannya yang begitu luas dan terbuka untuk dikombinasikan dengan sistem-sistem penunjang lainnya. Selain itu, hal yang perlu diperhatikan ialah jenis protokol yang akan menunjang integrasi sistem. Banyaknya variasi protokol yang bisa digunakan memunculkan rasa ragu dan bimbang bagi para pengembang teknologi ini perihal protokol mana yang lebih baik dan lebih cocok untuk diterapkan di tiap-tiap sistem yang akan dibangun. Akhirnya para pengembang tersebut mencoba mencari tahu serta membandingkan pola dan tingkat efektivitas tiap-tiap protokol penunjang. Tiap-tiap protokol memiliki peruntukannya masing-masing, hal ini terbukti dari perbedaan tingkat konektivitas, proses, dan hasil olahan data yang tersaji. Dalam Jurnal ini, akan dibahas komparasi antara dua tipe protokol yakni *HTTP* dan *MQTT*. Komparasi semacam ini berguna menambah wawasan dan pertimbangan bagi pengguna mengenai penggunaan protokol mana yang lebih tepat untuk digunakan di masing-masing kondisi sistem serta kelemahan dan kelebihan dari tiap-tiap protokol.

A. *Rumusan Masalah*

Berdasarkan pemaparan beberapa topik bahasan yang ada pada bagian pendahuluan, dapat kita rangkum dan susun pokok masalah yang akan dibahas pada jurnal ini, hal-hal tersebut antara lain:

1. Apa pengertian dari teknologi perangkat pintar (*IoT*/ *Internet of Things*)?
2. Apa saja manfaat yang bisa teknologi *IoT* berikan bagi kehidupan sehari-hari?
3. Apa saja kelemahan *IoT*?
4. Apa fungsi dari *RabbitMQ*?
5. Apa saja letak perbedaan antara protokol *HTTP* dengan protokol *MQTT*?
6. Bagaimana proses lalu lintas data menggunakan *message broker*?
7. Bagaimana peran *docker* selaku media *containerize*?
8. Bagaimana implementasi sistem *IoT* pada *server* berbasis *cloud*?

B. *Batasan Masalah*

Penentuan beberapa batasan masalah dibuat untuk menghindari adanya pelebaran pokok bahasan yang berpotensi menimbulkan penyimpangan materi dan fokus utama informasi jurnal, penulis juga lebih bisa mengarahkan pembaca agar tidak keluar dari jalur pembahasan sehingga tujuan penelitian yang telah disusun dapat tercapai dan tepat pada sasaran yang dituju. Terdapat 3 batasan masalah yang telah ditentukan, yakni:

1. Ruang lingkup pembahasan hanya meliputi analisis dan komparasi antara dua protokol yakni protokol *HTTP* dan *MQTT*.
2. Komparasi protokol yang dilakukan meliputi: model protokol, kecepatan pengiriman data, ukuran pesan yang dikirimkan, serta penggunaan optimal untuk masing-masing protokol.
3. Penggunaan sistem-sistem penunjang (*RabbitMQ*, *docker container*) berguna menunjang sistem utama (*cloud-IoT*).

C. Tujuan Penelitian

Dalam pembuatan jurnal ini, terkandung sasaran atau keinginan yang ingin dicapai oleh penulis melalui penelitian yang dilakukan. Target atau tujuan tersebut antara lain untuk mengetahui tingkat efektivitas antara dua protokol yaitu *HTTP* dan *MQTT* terhadap implementasi sistem *cloud-IoT* yang didukung oleh beberapa sistem penunjang. Harapan dari tercapainya tujuan tersebut yaitu pengembang sistem dapat lebih memaksimalkan kinerja teknologi *IoT* dengan pemilihan protokol yang tepat, yang dapat meningkatkan kualitas *development* sistem.

Disekeliling kita sudah banyak sekali sistem yang memanfaatkan teknologi *IoT*, tugas penulis ialah membuka cakrawala pembaca dengan melakukan pengenalan terhadap teknologi *IoT* lewat tulisan jurnal bermanfaat seperti ini.

D. Manfaat Penelitian

Terdapat 2 jenis manfaat yang diperoleh oleh penulis maupun pembaca dari hasil penelitian ini, hal tersebut antara lain:

1. Manfaat Teoritis

Hasil dari penelitian penulis dalam memberikan tulisan/ informasi melalui jurnal ini diharapkan dapat mencapai manfaat teoritis, yakni menjadi landasan/ dasar ilmu pengetahuan bagi para pengembang sistem untuk menentukan jenis protokol yang cocok dan efektif untuk diterapkan pada sistem *IoT* berbasis *cloud* yang mereka kembangkan nantinya.

2. Manfaat Praktis (*Practical*)

a. Bagi pembaca (masyarakat)

Pada bagian ini, jurnal bacaan dapat memberikan referensi/ wawasan bagi masyarakat diluar sana yang sama sekali belum mengenal apa itu teknologi *Internet of Things* (IoT), apa saja fungsi-fungsinya, apa tujuan dan kemudahan apa yang ditawarkan dari dibangunnya sistem tersebut, serta teknologi apa yang yang berkaitan dengan pengembangan sistem perangkat pintar ini.

b. Bagi penulis

Dalam hal ini, penulis sekaligus peneliti dapat turut berkontribusi secara langsung dalam memajukan sumber informasi (bacaan) terkait pemilihan jenis protokol yang tepat untuk teknologi IoT. Hal tersebut meliputi bentuk komparasi protokol, yang meliputi model dan jenis lalu lintas datanya, beberapa fitur yang ditawarkan dari masing-masing protokol, serta kelebihan dan kelemahan protokol terkait yang disajikan melalui penulisan jurnal ini.

II. TINJAUAN PUSTAKA

Teknologi IoT (*Internet of Things*) kini kian memperluas manfaat dari komunikasi jaringan internet sehingga benda-benda disekitar kita yang dihubungkan dengan satu jaringan yang sama dan dapat dikontrol untuk melakukan suatu perintah, melakukan monitoring,

ISSN: 2614-8404

pelacakan data, mengumpulkan data, ataupun mengirim data dari jarak cukup jauh (tidak bergantung pada ruang dan waktu).

Namun, tentu saja teknologi ini tidak sepenuhnya sempurna, kemampuan pemecahan masalah yang dilakukan masih memiliki keterbatasan yang perlu kita benahi. Kendala-kendala yang biasa kita jumpai dalam teknologi ini yakni, keterbatasan dalam hal kapasitas penyimpanan data serta kemampuan komputasi sistem. Hal ini bisa saja menciptakan keraguan bagi pengguna teknologi *IoT* yang menganggap remeh tentang kinerja sistem, kehandalan layanan, dan keamanan terhadap data pengguna yang dikelola oleh sistem. Selain itu, protokol HTTP yang mulanya dipercaya dalam menangani lalu lintas data antara perangkat IoT dengan *server*, ternyata memiliki sistemasi yang kurang efektif untuk digunakan karena sistem kerja yang cukup rumit untuk tiap kali pengiriman *request*. Semisal, pada saat mengirimkan *request* pada perangkat IoT, lalu lintas data yang melewati protokol HTTP diharuskan untuk mengalami proses *encode* ataupun *decode* dengan mencantumkan ukuran *header* cukup besar, hal ini mempengaruhi ukuran ruang memori yang terpakai, padahal ruang memori untuk perangkat ini hanya tersedia terbatas. Hal ini butuh perhatian lebih dalam lagi supaya masalah efektivitas dari protokol yang digunakan, tidak menjadikan hilangnya fungsi utama dari teknologi IoT itu sendiri. Untuk mengatasi kelemahan teknologi tersebut, perangkat IoT dapat diintegrasikan dengan sistem handal lain yang memiliki kapasitas penyimpanan dan kemampuan komputasi yang lebih baik. Salah satunya adalah sistem komputasi berbasis *cloud*. Menurut seorang ahli, definisi *cloud computing* dapat dipahami sebagai sebuah model komputasi baru dimana sumber daya komputasinya dapat dikonfigurasi sesuai kebutuhan pengguna secara mudah melalui jaringan internet (Zhang, et., all, 2010).

Dengan adanya sistem *cloud*, pengguna dapat menyimpan data dengan kapasitas penyimpanan yang sangat besar (awan), mudah untuk diintegrasi, serta lebih mudah dalam mengakses data kapan saja dan dimana saja karena tidak terikat media penyimpanan fisik. Protokol *MQTT* atau *Message Queue Telemetry Transport*, menggunakan pola *publish*/ *subscribe* yang dirancang terbuka, ringan, dan mudah untuk diimplementasikan. Pola *publish*/*subscribe* memiliki kelebihan yang disebut *decouple*, maksudnya baik pengirim maupun penerima pesan tidak berkomunikasi secara langsung. Hal ini memungkinkan *sender* dapat mengirim pesan kepada *receiver* kapanpun dibutuhkan, karna sudah ada *MQTT broker* yang menjamin jalur distribusi pesannya. *Topic* atau *routing information* terdapat pada *broker MQTT*. Tiap kali *receiver* ingin menerima pesan, maka dapat langsung men-*subscribe topic* tertentu, dan selanjutnya *broker* akan mengirimkan semua pesan yang sesuai dengan *topic* tersebut. Dengan kata lain, komunikasi yang terjadi bukan antara *sender* dengan *receiver*, melainkan antara *sender* dengan *broker* dan *receiver* dengan *broker*, yang mana cara berkomunikasi mereka menggunakan *topic*. Protokol *MQTT* memerlukan transportasi *byte code* dari *client* ke *server* atau *server* ke *client*. Protokol *transport* yang digunakan yakni TCP/IP. Protokol *transport* TCP/IP tidak hanya digunakan oleh *MQTT*, melainkan juga digunakan oleh *TLS* dan *WebSocket*. Hal ini dikarenakan jaringannya yang bersifat *connectionless*.

Berikut fitur-fitur yang ditawarkan oleh protokol *MQTT*:

1. Menggunakan jalur *transport* jaringan TCP/IP.
2. Adanya tiga level Qos (*Qualities of Service*) ketika menyampaikan pesan melalui protokol MQTT:

o"*At most once*", maksudnya pesan dikirim dengan upaya terbaik dari jaringan *transport* TCP/IP. Namun, kehilangan pesan atau duplikasi tetap dapat terjadi.

o"*At least once*", yakni dipastikan kehilangan pesan tidak terjadi (pesan pasti terkirim), namun duplikasi masih dapat terjadi

3. "*Exactly once*", yakni dipastikan pesan terkirim tepat satu kali. Messagging *transport* dari *payload* yang *agnostic* disertai isi.
4. metode *publish*/*subscribe message pattern* yakni menyediakan distribusi pesan dari satu *publisher* ke banyak *subsriber* dan adanya *decoupling* aplikasi.

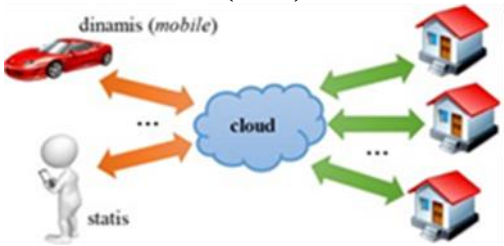
Pola *decoupling*/ *decouple* ini terbagi menjadi tiga bagian, yaitu *time decoupling*, *space decoupling*, dan *synchronization decoupling*.

Yang pertama adalah *Time decoupling*, artinya tiap-tiap subjek (pengirim maupun penerima) tidak diharuskan untuk aktif dalam waktu yang bersamaan.

Yang kedua adalah *space decoupling*, artinya masing-masing subjek (pengirim ataupun penerima) diharuskan untuk aktif dalam waktu bersamaan, namun baik pengirim ataupun penerima tidak mengetahui keberadaan dan identitas satu sama lain.

Yang terakhir adalah *synchronization decoupling*, artinya pengaturan pengiriman pesan antara pengirim maupun penerima tidak saling bergantung (*independent*).

Jenis protokol ini dapat menangani ribuan client dari jarak cukup jauh hanya dengan satu *server*. Protokol MQTT meminimalkan penggunaan bandwith jaringan, memiliki ukuran *header* yang sangat kecil (2 bytes) dan hemat kebutuhan sumber daya perangkat sehingga sangat cocok diterapkan pada teknologi IoT yang mengandalkan prinsip *machine to machine* (M2M).

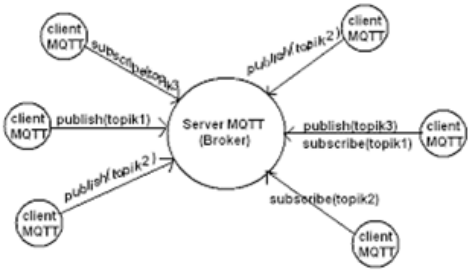


Gambar. 1 Ilustrasi implementasi *cloud system* pada teknologi IoT.

Ketika data dikirimkan menuju *cloud server*, fungsi RabbitMQ selaku *message broker* akan berjalan. Analogi paling sederhana yang dapat menggambarkan RabbitMQ adalah seperti kantor pos dan tukang pos. RabbitMQ ini akan menerima pesan dari pengirim/ *sender*/ *publisher* dan kemudian akan meneruskannya kembali (*forward* and *pushing*) kepada penerima/ *receiver*/ *subscriber*. Dari perilaku tersebut, dapat kita simpulkan bahwa pesan sendiri memiliki sifat *asynchronous* yang mana untuk menjalankan *behavior* tidak harus menunggu/ bergantung pada kesiapan *entity* lain. Sehingga dalam hal ini, pengirim tidak harus menunggu konfirmasi dari penerima perihal keberhasilan pesan terkirim. Semua pesan yang dikirim akan tertampung

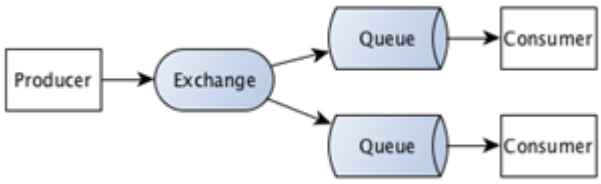
ISSN: 2614-8404

pada broker (tepatnya pada *queue*) sampai nantinya diteruskan pada tujuan, masing-masing pihak (pengirim dan penerima) dapat menjalankan aktivitas/ proses lain tanpa terhambat kesiapan ataupun konfirmasi dari masing-masing pihak. RabbitMQ ini berjalan pada *docker container*, maka dari itu *container* yang telah terisi oleh RabbitMQ harus terlebih dahulu terinstall pada sistem. Hal ini sangat memudahkan kita karena tidak perlu menginstall dan mengonfig dari awal segala sesuatunya, kita tinggal menginstall *docker*-RabbitMQ yang sudah siap pakai. Hal ini menjadi sebegitu praktis karna semisal kita ingin berpindah *server*, dan ingin RabbitMQ terinstall pada *server* tersebut maka kita tidak perlu repot-repot men-*download* dan mengonfig segalanya dari awal secara manual. Kita hanya perlu memindahkan *docker*-RabbitMQ yang telah siap pakai tersebut ke *server* yang baru. Selanjutnya pembahasan mengenai alur masuk data. Alur masuk data pada sistem adalah data akan masuk pada bagian *exchange*, bagian ini bertugas mengatur pembagian data ke masing-masing queue. Tiap-tiap *queue* akan mendistribusikan data ke consumer/ *receiver*/ *subscriber*.



Gambar. 2 Ilustrasi teknologi RabbitMQ dalam pendistribusian data.

Namun, karna umumnya RabbitMQ menggunakan protokol AMQP, sehingga untuk menggunakan protokol MQTT diperlukan *software* tambahan untuk menjalankan program (*plugin*) yang bernama *rabbitmq-plugins*.



Gambar 3. Ilustrasi alur *request data* menuju *server* dari beberapa perangkat pintar yang masing-masing menggunakan protokol MQTT. Semua *request* menuju *server* telah terlebih dahulu melewati *broker*.

III. METODOLOGI PENELITIAN

Berdasarkan dari beberapa objek masalah yang ditemui, diputuskan beberapa penggunaan metode penelitian yang cocok untuk diaplikasikan pada riset ini. Metodologi yang diterapkan dalam implementasi serta analisis sistem ini ialah metode kualitatif/ studi literasi dan metode kuantitatif. Maksud dari metode kualitatif yakni mengangkat sebuah hasil berdasarkan pencarian informasi dan data komparasi dari berbagai sumber aktual. Metode ini mengumpulkan data dalam bentuk naratif (deskripsi).

Setelah riset dilakukan, barulah ada penarikan jawaban sementara. Jawaban ini yang nantinya melewati fase pengujian dan penekanan bukti terkait. Fase pengujian berdasar pada pengalaman partisipan terkait dalam mencoba mengimplementasikan sistem. Semua informasi subjektif yang sebelumnya didapat akan dibuktikan benar atau tidaknya. Menganilisis hasil uji juga menjadi poin penting karna hasil uji yang sudah didapat perlu disusun rapih untuk nantinya memudahkan kita dalam melakukan metode kuantitatif. Setelah melewati fase pengujian dan sudah mendapatkan beberapa bukti objektif, barulah metode kuantitatif dilakukan. Metode ini memperkuat hasil/ bukti riset, pengujian, dan analisis melalui penggambaran tabel komparasi mengenai informasi yang diperoleh atas riset yang telah dilakukan.

IV. HASIL DAN PEMBAHASAN

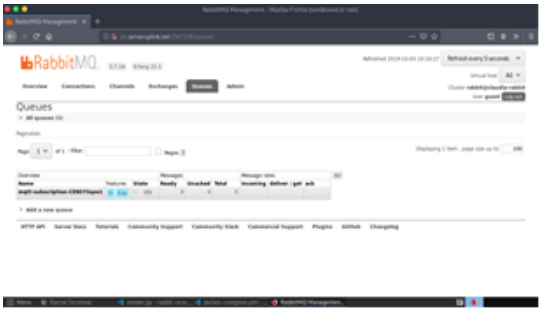
Dari berbagai studi literasi yang didapatkan, perlu adanya pengujian secara langsung oleh peneliti supaya keakuratan informasi serta data hasil dapat terbukti benar. Dibawah ini adalah beberapa rekam foto pada saat pengujian dilakukan:

Hal yang utama dilakukan ialah men-setup RabbitMQ (*broker-message*) pada *cloud-server*, tahap ini bertujuan untuk menciptakan *temporary place* dalam pendistribusian pesan. Caranya dengan mengakses alamat dari *cloud-server* dengan diikuti oleh *port* RabbitMQ, mengisi *username* dan *password* yang diminta, lalu klik *login*.

url: http://jts.serveruplink.net
port RabbitMQ: 15672
username: *guest*
password: *guest*

Gambar 4 Halaman *login* RabbitMQ *services* pada *cloud-server*.

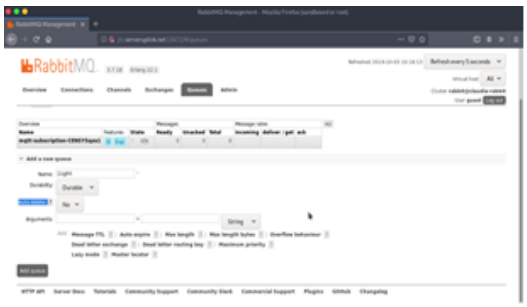
Selanjutnya membuat *queue* baru dengan cara mengklik *queue section* lalu pilih *add queue* dan isi nama dari *queue* yang anda buat.



Gambar 5 Tampilan RabbitMQ dalam membuat *queue*.

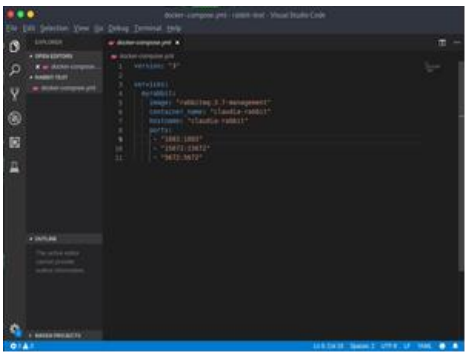
Selanjutnya buat *exchange* baru pada *exchange section* dengan klik *add new exchange*, lalu tulis nama *queue* yang telah dibuat, lalu klik *bind* untuk menyambungkan koneksi keduanya.

ISSN: 2614-8404



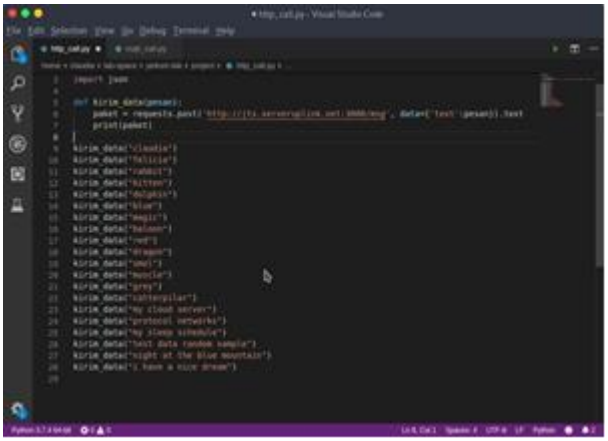
Gambar 6 Tampilan RabbitMQ dalam membuat *exchange*.

Buatlah konfigurasi pada *docker-compose* yang nantinya akan diletakkan pada *cloud-server*.



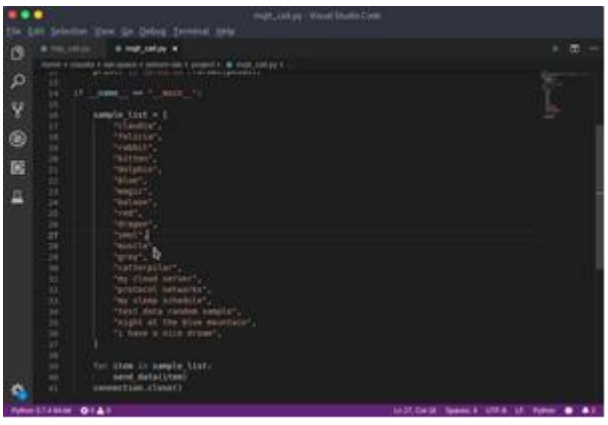
Gambar 7. Contoh *code* konfigurasi *docker response*.

Berikut tampilan sekilas kode untuk mengirim data dari protokol HTTP.



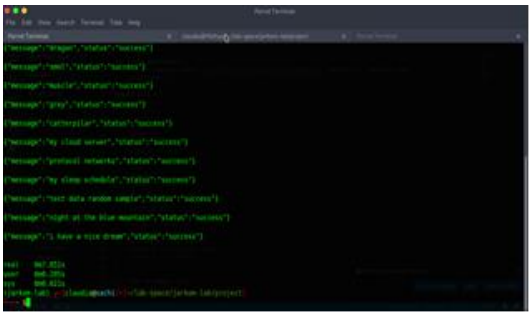
Gambar 8. Contoh *code* untuk mengirim data dari protokol HTTP.

Berikut adalah tampilan sekilas kode untuk mengirim data dari protokol MQTT.

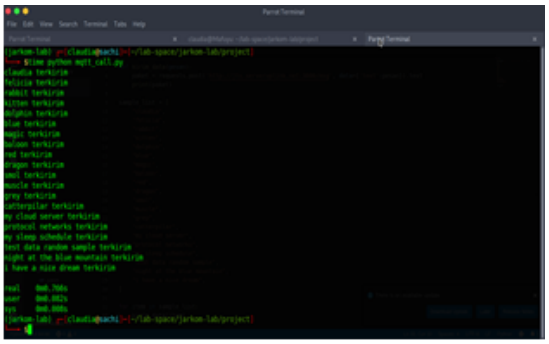


Gambar 9. Contoh *code* untuk mengirim data dari protokol MQTT.

Berikut dibawah ini adalah komparasi lama waktu request data yang dihasilkan protokol HTTP dan MQTT.

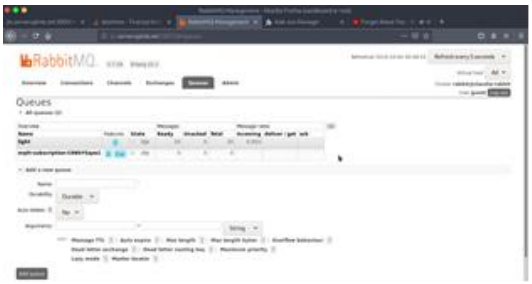


Gambar 10. Lama waktu proses dari protokol HTTP.



Gambar 11. Lama waktu proses dari protokol MQTT.

Hasil *request* data yang tertera pada *section total* (tertera angka 20) sebagai *logging* pada laman RabbitMQ *services*.



Gambar 12.Hasil *request* data pada RabbitMQ.

Dalam hal ini, penguji menggunakan bahasa pemrograman python sebagai bahasa *server-side* dan menulisnya pada *text editor vscode*. Sebelum menjalankan *script-script* diatas, *server* RabbitMQ pada *server* telah terlebih dahulu diaktifkan dan *firewall* pada *server* harus di non-aktifkan terlebih dahulu. Dari data diatas, terbukti kecepatan *request* per data dari 20 data yang tersedia untuk masing-masing protokol, ternyata menghasilkan waktu yang berbeda. Protokol MQTT terbukti lebih cepat (berada di angka 0.7 detik) dibanding protokol HTTP (berada di angka 7.8 detik). Jadi dapat kita simpulkan bahwa perbandingan diantara keduanya adalah 1:9. Protokol MQTT 9 kali lebih cepat memproses data dibanding protokol HTTP.

Tidak hanya dari segi kecepatan, berikut perbedaan protokol HTTP dan MQTT dari segi model protokol, ukuran pesan yang dikirimkan, serta penggunaan optimal untuk masing-masing *protocol*:

1. Model Protokol

ISSN: 2614-8404

HTTP -> protokol dokumentaris
MQTT -> protokol datasentris

2. Ukuran Pesan yang Dikirimkan
   HTTP -> diatas 2 MB
   MQTT -> dibawah 2 MB
3. Bandwith Penunjang
   HTTP -> besar (resource intensive)
   MQTT -> terbatas/ kecil maupun besar
4. Penggunaan Optimal
   HTTP -> sistem dengan data skala besar
   MQTT -> teknologi IoT (microcontroller)
5. Library Pengiriman Data
   HTTP -> menggunakan request
   MQTT -> menggunakan pika

V. PENUTUP

Setelah dilakukannya penelitian melalui berbagai sumber informasi serta implementasi secara langsung, maka penulis dapat mengambil inti/ simpulan sebagai berikut :

Jurnal ini telah merangkum protokol *transfer data* yang digunakan dalam IoT. IoT diharapkan dapat diterapkan ke berbagai bidang/ aplikasi sebagai infrastruktur sosial. Namun, untuk menyebarkan IoT secara luas, Diperlukan protokol komunikasi ringan dan efektif. Melalui jurnal ini telah dijelaskan bahwa arsitektur MQTT Memiliki pola yang sesuai dan sangat cocok untuk diterapkan oleh IoT. Perbandingan telah dibuat antara protokol HTTP dalam kategori protokol warisan dan MQTT dalam kategori protokol pembaharu yang ringan dan cepat. Melalui penulisan jurnal ini terbukti bahwa protokol MQTT berkinerja lebih baik, efektif, dan cepat daripada HTTP. Perbandingannya mencapai 9:1, yakni protokol MQTT 9 kali lebih cepat dibanding protokol HTTP. Selain itu, jurnal ini bertujuan mengusulkan pendekatan untuk meningkatkan penggunaan protocol MQTT.

Untuk penelitian dan penulisan selanjutnya, penulis menyarankan beberapa poin yang diharapkan nantinya dapat diwujudkan seiring dengan perkembangan media informasi ini, beberapa saran yang diusulkan antara lain:

1. Pada penelitian selanjutnya diharapkan untuk dapat mengembangkan *cloud* sistem *server* dengan skala yang lebih besar, kompleks, terintegrasi. contohnya menggunakan layanan komputasi *Amazon Web Services* (AWS). *Platform* tersebut menawarkan layanan *cloud-computing* yang lengkap dengan teknologi yang terintegrasi dengan baik.
2. Mampu membuat media informasi sebagai hasil penelitian komparasi teknologi lainnya. Tidak hanya protokol HTTP dan MQTT saja, dapat juga dibuat pembahasan mengenai protokol AMQP, CoAP, dan lain sebagainya.
3. Adanya fitur tambahan yakni *logging*/ *error tracker* pada *server* untuk mendeteksi terjadinya *error* atau kesalahan *transport data*, guna memudahkan pada saat *troubleshooting*.

DAFTAR PUSTAKA

[1] Abdurohman, A, 2017. Messaging dengan message broker Rabbitmq dan Java menggunakan Spring Framework. In *Computer Network*, p.1.

[2] Atmoko*, R. A., & , R Riantini, and M. K. H,. 2017. IoT real time data acquisition using MQTT protocol. *IoT Real Time Data Acquisition Using MQTT Protocol*, *853*(IOP Publishing), 1–6.

[3] Boronat, J. E. L. ; J. C. C. ; C. C. ; P. M. ; M. P. ; P., 2015. Handling mobility in IoT applications using the MQTT protocol. In Handling mobility in IoT applications using the MQTT protocol. (pp. 1–10).

[4] Geeknesia., 2016. Menggunakan HTTP atau MQTT? (p. 1).

[5] Javed, A., & Javed, A., 2016. IoT Platforms. In *Building Arduino Projects for the Internet of Things*. https://doi.org/10.1007/978-1-4842-1940-9_12.

[6] Manohar, H. L., & Reuban Gnana Asir, T., 2018. Data consumption pattern of MQTT protocol for IoT applications. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-981-10-7635-0_2

[7] Naik, N., 2017. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*. https://doi.org/10.1109/SysEng.2017.8088251

[8] Satria, G. O., Satrya, G. B., & Herutomo, A., 2015. Implementasi Protokol MQTT Pada Smart Building Berbasis OpenMTC. *E-Proceeding of Engineering*.

[9] Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P., 2015. Secure MQTT for Internet of Things (IoT). Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015. https://doi.org/10.1109/CSNT.2015.16

[10] Vandikas, K., & Tsiatsis, V., 2014. Performance evaluation of an iot platform. *Proceedings - 2014 8th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST 2014*. https://doi.org/10.1109/NGMAST.2014.66

[11] Yokotani, T., & Sasaki, Y., 2017. Comparison with HTTP and MQTT on required network resources for IoT. *ICCEREC 2016 - International Conference on Control, Electronics, Renewable Energy, and Communications 2016, Conference Proceedings*. https://doi.org/10.1109/ICCEREC.2016.7814989