JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 10
No. 1
April 2024

Efisiensi Pertukaran Data *Client-Server* menggunakan *Web Socket* pada Perangkat Berbasis *Internet of Things*

Kristian Adi Nugraha[#1]

[#] *Informatika, Universitas Kristen Duta Wacana*
Jalan Dr. Wahidin Sudirohusodo Nomor 5-25 Yogyakarta 55224
[1]adinugraha@ti.ukdw.ac.id

Abstrak— ***Internet of Things* (*IoT*) banyak digunakan dengan tujuan membantu mempermudah hidup masyarakat dalam wujud perangkat-perangkat yang dapat bekerja secara otomatis. Perangkat *IoT* secara umum dapat dipantau atau dikendalikan menggunakan perangkat lain seperti komputer atau telepon seluler melalui perantara *server* yang dikenal sebagai *Application Programming Interface* (API). Umumnya, protokol yang digunakan untuk pertukaran data antara *client* dan *server* adalah *Message Queuing Telemetry Transport* (MQTT). Namun MQTT memiliki beberapa kelemahan di sisi kehandalan dan keamanan data, sehingga tidak dapat digunakan pada segala jenis kondisi dan situasi. Sebagai alternatif, terdapat protokol *Hypertext Transfer Protocol* (HTTP) yang dapat mengakomodasi kekurangan tersebut. Namun HTTP memiliki kelemahan di mana *server* tidak dapat mengirimkan data kepada *client* apabila *client* tidak mengirimkan *request* terlebih dahulu. Sementara pada perangkat IoT, data seharusnya bisa dikirimkan oleh kedua belah pihak. Terdapat satu buah protokol lain yang dapat menjadi solusi dari kedua kebutuhan tersebut, yaitu *Web Socket*. Penelitian ini bertujuan untuk mengimplementasikan arsitekur *client-server* berbasis *Web Socket* untuk melakukan pertukaran data antara perangkat *IoT* dengan *server*. Hasil pengujian menunjukkan bahwa protokol *Web Socket* memiliki nilai *latency* yang paling rendah di antara protokol yang lain, yaitu dengan nilai rata-rata antara 10,75 sampai dengan 23,29 milidetik. Rentang tersebut termasuk kategori yang baik, sehingga dapat disimpulkan bahwa *Web Socket* dapat digunakan sebagai alternatif pengganti MQTT.**

Kata kunci*— API, HTTP, *Internet of Things*, MQTT, *Web Socket

I. PENDAHULUAN

Internet of things (IoT) adalah sebuah teknologi yang memberi nilai tambah pada sebuah benda atau perangkat, yang semula hanya bisa digunakan secara konvensional, menjadi dapat dikendalikan dari jarak jauh melalui jaringan *internet*. Teknologi IoT umumnya diaplikasikan dalam bentuk sistem tertanam atau *embedded system*. Pada contoh lain, IoT juga banyak diimplementasikan pada perangkat-perangkat berukuran kecil atau berbentuk robot. Karena biaya implementasi yang semakin terjangkau, IoT banyak dikembangkan selama beberapa tahun terakhir untuk berbagai macam bidang, misalnya keamanan [1], pertanian [2], dan kesehatan [3]. Berdasarkan fleksibilitas dan semakin terjangkaunya harga perangkat IoT, maka teknologi IoT diharapkan dapat semakin banyak diimplementasikan dan digunakan untuk membantu mempermudah masyarakat saat ini dalam menjalankan kehidupan sehari-hari.

Salah satu implementasi dari IoT yang saat ini cukup banyak dikembangkan adalah dalam wujud kendaraan yang berfungsi untuk menyelesaikan berbagai tugas guna memecahkan sebuah masalah. Misalnya mobil cerdas yang dapat berjalan secara otomatis dan dapat dipantau melalui perangkat khusus dari jarak jauh melalui perantara jaringan *internet* [4]. Mobil tersebut dapat digunakan untuk membantu orang-orang yang tidak memiliki kemampuan menyetir atau orang-orang berkebutuhan khusus agar tetap berkendara secara mandiri dengan mobil. Mobil tersebut juga dapat melakukan perhitungan terhadap beberapa faktor seperti trafik kemacetan lalu lintas yang didapatkan dari *server* agar mobil dapat memperhitungkan rute yang paling efisien. Contoh lain yang serupa, yaitu berupa kendaraan dengan teknologi IoT sehingga dapat berjalan secara otomatis. Misalnya mobil cerdas dengan teknologi IoT, *Edge Intelligence*, koneksi 5G, dan Blockchain [5]. Dengan adanya teknologi tersebut, maka terdapat banyak orang yang tidak dapat mengendarai mobil, dapat terbantu karena mobil dapat berjalan secara otomatis. Kedua contoh tersebut merupakan penerapan teknologi IoT yang membutuhkan pertukaran data dengan frekuensi yang cukup tinggi secara *real-time* dan membutuhkan waktu transmisi yang cepat, karena hal tersebut terkait dengan faktor keselamatan (*safety*) pengendara maupun lingkungan di sekitarnya. Sehingga apabila terdapat jeda waktu (*delay*) yang cukup lama atau *latency* yang tinggi dalam melakukan transmisi data, maka data tersebut sudah tidak dapat digunakan karena kondisi lapangan yang sudah berubah dalam waktu singkat. Contoh kasus lain yang memerlukan transmisi data secara *real-time* adalah pada sistem peringatan bencana [6], misalnya untuk bencana

banjir [7], di mana sistem tersebut harus mampu mengirimkan data secara *real-time*, karena apabila terlambat beberapa detik maka akan dapat berakibat fatal.

Penelitian ini mencoba untuk melakukan eksperimen terkait pertukaran data antara perangkat IoT dan *server* dengan menggunakan koneksi berbasis *Web Socket*. Web *Socket* merupakan jenis protokol koneksi penuh tanpa putus, sehingga proses transmisi data dapat dilakukan dengan cepat. Hal ini menjadi penting ketika dibutuhkan proses pengiriman data yang relatif cepat dengan *latency* yang rendah agar data tersebut dapat segera diproses lebih lanjut. Umumnya, perangkat IoT melakukan pertukaran data antar perangkat dengan menggunakan protokol *Message Queuing Telemetry Transport* atau MQTT [8]. Namun protokol MQTT memiliki beberapa kekurangan, salah satunya adalah tidak adanya mekanisme pengamanan seperti pada protokol yang lain. Hal ini menjadi cukup berbahaya apabila data yang dikirimkan merupakan data sensifit, misalnya data informasi pribadi atau data terkait transaksi keuangan. Apabila data tersebut disadap oleh pihak tertentu, maka pemilik data akan sangat dirugikan. Kekurangan lain dari MQTT adalah tidak adanya pesan kembalian (*response*) saat sebuah perangkat mengirimkan data ke perangkat lain, sehingga sulit untuk diketahui apakah pesan tersebut berhasil sampai ke tujuan atau tidak. Hal ini membuat aktivitas pengiriman data dengan menggunakan protokol MQTT kurang dapat diandalkan, karena sulit memastikan apakah data tersebut telah dapat diterima dengan baik atau tidak. Oleh karena itu, MQTT lebih banyak digunakan untuk keperluan transmisi data yang tidak terlalu krusial, sehingga apabila data tersebut bocor atau tidak tersampaikan dengan baik, maka tidak terlalu berdampak besar bagi pihak pengirim maupun penerima data.

II. TINJAUAN PUSTAKA

Internet of Things (IoT) adalah teknologi yang banyak dimanfaatkan untuk meningkatkan efisiensi dalam aktivitas sehari-hari [9], salah satu contohnya adalah implementasi kota pintar atau *smart city* [10]. Pada *smart city*, terdapat sebuah komputer berukuran mikro beserta sensor di dalamnya yang dipasang pada beberapa titik fasilitas publik, misalnya seperti kamera pemantau, lampu APILL (Alat Pemberi Isyarat Lalu Lintas), dan sepanjang jalan raya. Dengan menerapkan teknologi IoT, perangkat-perangkat tersebut memiliki kemampuan tambahan untuk dipantau dari jarak jauh melalui perangkat *smartphone* atau komputer. Dengan demikian, apabila terdapat kejadian seperti kecelakaan dan sejenisnya, maka dapat lebih cepat untuk ditangani. Contoh lain penerapan teknologi IoT adalah perangkat untuk melakukan monitoring kondisi udara dengan tingkat skalabilitas yang mampu disesuaikan dengan kebutuhan [11]. Melalui perangkat tersebut, maka tingkat polusi udara pada beberapa sektor publik seperti rumah sakit, sekolah, atau tempat umum yang lain dapat dengan cepat terdeteksi.

IoT mewajibkan sebuah perangkat terhubung ke jaringan internet supaya dapat melakukan pertuakaran data dengan perangkat lainnya [12]. Supaya dapat melakukan komunikasi dua arah antar perangkat, dibutuhkan komputer yang berperan sebagai pihak perantara bernama *web service* yang memiliki wujud berupa *Application Programming Interface* (API) [13]. API memiliki banyak sekali jenis, salah satu jenis API yang banyak diimplementasikan untuk keperluan *web service* adalah RESTful. RESTful dapat menghasilkan performa yang cukup baik dengan kebutuhan sumber daya yang cukup ringan [14]. Sedangkan untuk keperluan jenis format data, jenis yang paling banyak dimanfaatkan untuk keperluan RESTful API adalah format data JSON [15]. Hal ini disebabkan karena format data JSON memiliki ukuran relatif kecil jika dibandingkan dengan format lain seperti XML. Namun pada perangkat IoT, umumnya API dibangun dengan protokol bernama *Message Queuing Telemetry Transport* (MQTT) [16]. Jika dibandingkan dengan protokol HTTP pada umumnya, kelebihan MQTT dalam melakukan transmisi data adalah sifatnya yang ringan sehingga tidak terlalu menghabiskan sumber daya (*resource*) terlebih saat terhubung dengan banyak perangkat. Selain itu, MQTT menggunakan skema *publisher* dan *subscriber*, sehingga pengiriman data bisa diinisiasi oleh semua pihak, tidak seperti protokol HTTP di mana data harus diinisiasi oleh *client* dalam bentuk *request* kepada *server*. Hal tersebut yang menyebabkan MQTT cukup populer digunakan pada banyak aplikasi dibandingkan dengan HTTP [17]. Namun salah satu kekurangan utama dari MQTT adalah tidak adanya mekanisme pengamanan seperti pada protokol lain pada umumnya [18], sehingga terlalu beresiko apabila digunakan untuk mengirimkan data yang bersifat sensitif. Kekurangan yang lain, karena menggunakan mekanisme *publish* dan *subscribe*, maka pihak pengirim data hanya bertugas untuk mengirimkan data tanpa tahu data tersebut diterima dengan baik atau tidak [19].

Berdasarkan pemaparan tersebut, maka dapat protokol MQTT tidak selalu dapat digunakan untuk keperluan tertentu saat mengimplementasikan teknologi IoT. Di sisi lain, apabila diimplementasikan dengan menggunakan protokol HTTP, maka proses pengiriman data hanya dapat dilakukan satu arah dari *client* kepada *server*, karena *server* tidak dapat mengirimkan data kepada *client* apabila tidak diminta terlebih dahulu oleh *client*. Oleh karena itu, terdapat protokol lain yang tetap dapat mengirimkan data dari dua arah, namun tetap dapat memperhatikan sisi keamanan, yaitu protokol *Socket*. *Socket* atau *Web Socket* merupakan sebuah protokol untuk pengiriman data dengan waktu yang singkat [20] dan *latency* yang sangat rendah, oleh karena itu *Web Socket* banyak digunakan untuk keperluan transmisi data yang bersifat *real-time*. Penelitian ini bertujuan untuk menguji protokol *Web Socket* untuk keperluan transmisi data antara perangkat berbasis IoT dan *server* sebagai pusat penyimpanan data. Sehingga nantinya dapat disimpulkan apakah *Web Socket* dapat digunakan sebagai alternatif pengganti MQTT yang selama ini paling banyak digunakan untuk implementasi perangkat IoT.

A. *Internet of Things*

Internet of Things (IoT) adalah sebuah konsep yang memiliki tujuan untuk mengubah obyek biasa yang ada di sekitar kita menjadi memiliki fungsi tambahan untuk melakukan pertukaran data dengan perangkat lain [21]. Dengan fungsi pertukaran data yang dimiliki, perangkat IoT akan memiliki manfaat lebih, misalnya dapat dilakukan pengaturan untuk otomatisasi atau dipantau dari jarak yang cukup jauh [22, 23]. Penerapan IoT umumnya melibatkan beberapa modul tambahan seperti sensor [24, 25], motor, dan peralatan komunikasi untuk mengirimkan data ke perangkat lain [26]. Dengan adanya IoT diharapkan dapat mempermudah hidup masyarakat dalam melakukan aktivitas sehari-hari.

B. *Arsitektur Client-Server*

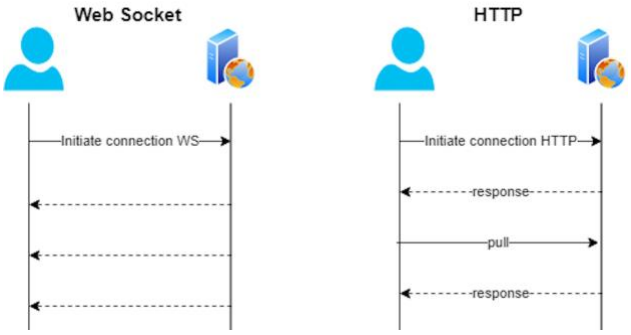
Client-Server merupakan salah satu jenis arsitektur perangkat lunak yang banyak diimplementasikan untuk keperluan pembangunan API dalam rangka pembuatan sebuah sistem yang mendukung berbagai macam jenis *platform* [27]. Dengan mengimplementasikan arsitektur *client-server*, maka seluruh data dapat tersimpan secara terpusat pada sebuah *server*, sehingga jika seorang pengguna hendak menggunakan dua atau lebih perangkat yang berbeda, maka seluruh perangkat tersebut tetap dapat melakukan sinkronisasi data antara satu dengan yang lain [28]. Salah satu jenis API yang banyak digunakan saat ini adalah RESTful API merupakan contoh penerapan arsitektur *client-server* yang banyak diimplementasikan karena faktor kemudahan dan penggunaan sumber daya yang tidak terlalu besar [29, 30]. Pada penelitian ini, *server* yang digunakan berupa komputer berbasis Linux sedangkan *client* adalah perangkat NodeMCU seperti ditunjukkan pada gambar 1.

Gambar. 1 Arsitektur pengujian *web socket*

C. *Web Socket*

Socket atau *Web Socket* merupakan sebuah protokol jaringan yang hampir serupa dengan protokol HTTP. Namun yang membedakan adalah koneksi pada protokol *Socket* bersifat permanen dan tidak akan putus meskipun *server* sudah mengirimkan response kepada *client* [31]. Selain itu, jumlah antara *request* dan *response* pada protokol *Socket* tidak harus sama, sehingga sangat memungkinkan apabila *server* mengirimkan data ke *client* tanpa *client* mengirimkan *request* terlebih dahulu seperti yang ditunjukkan pada gambar 2.

Karena memiliki karakteristik koneksi yang tidak terputus, maka protokol *Socket* memiliki waktu pengiriman data yang lebih cepat dibandingkan dengan protokol HTTP. Dengan demikian, dapat disimpulkan bahwa protokol *Socket* cocok digunakan untuk pembuatan sebuah sistem yang membutuhkan pengiriman data secara aktual atau *real-time* karena memiliki *latency* yang relatif lebih rendah [32].



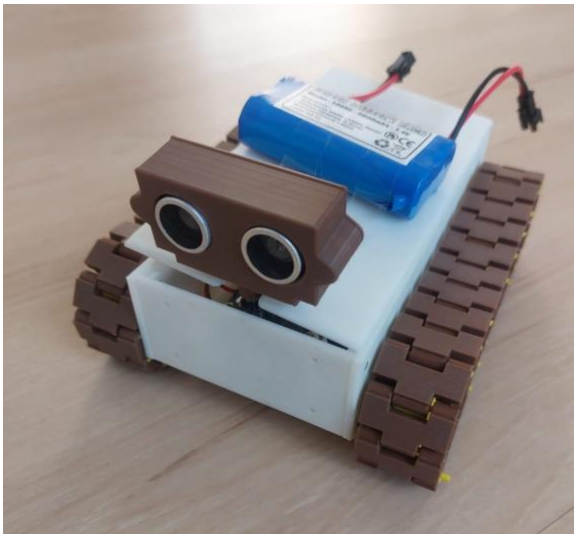
Gambar. 2 Perbedaan alur pengiriman *request* dan *response* pada *web socket* dan HTTP

III. METODE PENELITIAN

Penelitian dilakukan dengan menggunakan perangkat Micro-controller Unit (MCU) dengan seri NodeMCU ESP32 seperti ditunjukkan pada gambar 3. NodeMCU ESP32 merupakan sebuah perangkat yang banyak digunakan untuk membangun perangkat-perangkat berbasis IoT karena harganya yang terjangkau dan mudah didapatkan. Namun meskipun demikian, perangkat NodeMCU ESP32 tetap memiliki fitur yang lengkap, salah satunya adalah modul WiFi agar perangkat tersebut dapat terhubung melalui jaringan internet dan melakukan pertukaran data dengan perangkat lain seperti mikrokontroler atau dengan *server*.

Gambar. 3 Contoh perangkat NodeMCU ESP32

Tahap awal penelitian dimulai dengan membangun sebuah program dari kedua sisi arsitektur, yaitu pada bagian *client* dan pada bagian *server* seperti yang ditunjukkan pada gambar 1. Pada bagian *client*, yaitu pada perangkat ESP32, program dibangun dengan menggunakan bahasa pemrograman C. Sedangkan pada sisi *server*, program dibangun dengan menggunakan bahasa pemrograman Javascript (NodeJS). Untuk keperluan simulasi pengujian, perangkat ESP32 akan ditanamkan pada purwarupa kendaraan berukuran kecil seperti ditunjukkan pada Gambar 4. Kendaraan tersebut akan berjalan berputar di dalam sebuah ruangan sambil mengirimkan data ke *server* secara berkala. *Server* yang digunakan pada penelitian ini adalah *server* berbasis Linux yang terhubung melalui jaringan internet.



Gambar. 4 Purwarupa kendaraan berbasis IOT yang digunakan untuk pengujian

Data yang dikirimkan memiliki beberapa variasi ukuran, yaitu 1, 10, 100, dan 1000 karakter. Karena perangkat IoT umumnya hanya mengirimkan data-data sederhana, biasanya berupa nilai hasil pemindaian sensor, maka jumlah karakter maksimal pada pengujian ini dibatasi sebanyak 1000 karakter atau setara dengan 1 *kilobyte* data (dengan asumsi menggunakan jenis *encoding* UTF8). Untuk setiap parameter pengujian, purwarupa kendaraan akan mengirimkan sebanyak masing-masing 100 data dengan protokol *Web Socket*, 100 data dengan protokol HTTP, dan 100 data dengan protokol MQTT. Dengan demikian, maka terdapat 12 kali pengujian yang dilakukan seperti yang ditunjukkan pada tabel I.

TABEL I
SKEMA PARAMETER PENGUJIAN

Pengujian ke-	Ukuran (byte)	Jenis Protokol	Jumlah Sample
1	1	WebSocket	100
2	1	HTTP	100
3	1	MQTT	100
4	10	WebSocket	100
5	10	HTTP	100
6	10	MQTT	100
7	100	WebSocket	100
8	100	HTTP	100
9	100	MQTT	100
10	1000	WebSocket	100
11	1000	HTTP	100
12	1000	MQTT	100

Setiap data akan memiliki stempel waktu (*timestamp*) terkait kapan data tersebut dikirim dan kapan data tersebut sampai ke tujuan. Dari seluruh data yang berhasil disimpan, akan dianalisa lebih lanjut untuk dievaluasi dan diambil kesimpulan terkait performa dari masing-masing protokol. Pada protokol *Web Socket* dan HTTP, perhitungan waktu dihitung sejak data dikirimkan oleh perangkat IoT (*client*), kemudian diterima oleh *server* dan dikembalikan kepada *client* dalam bentuk respon. Sedangkan pada protokol MQTT agak sedikit berbeda karena protokol tersebut menggunakan konsep *publisher* dan *subscriber*, sehingga *latency* dihitung saat *publisher* mengirimkan data ke *server broker* dan diterima oleh para *subsciber*. Seluruh proses pengujian dilakukan dengan menggunakan jaringan internet bertipe *broadband* dengan kecepatan *bandwidth* rata-rata sebesar 30 megabit per detik.

Terdapat tiga kategori waktu respon dari sebuah API untuk menilai apakah waktu yang dihasilkan dapat terbilang cukup baik atau tidak [33] seperti yang ditunjukkan pada Tabel II.

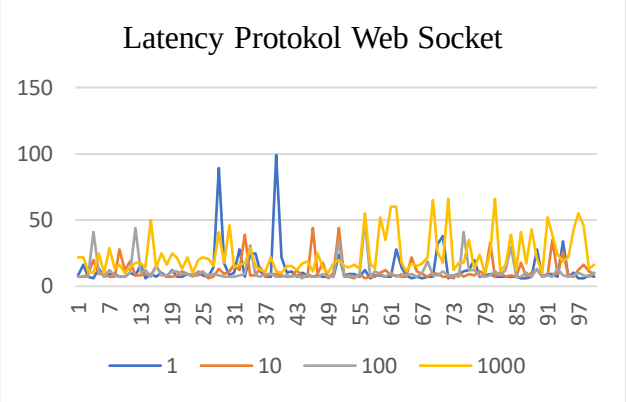
TABEL II
REFERENSI WAKTU RESPON API

Waktu	Keterangan
kurang dari 0,1 detik	Pengguna merasa sistem dapat merespon secara instan dan tidak ada interupsi
0,1 sampai 1 detik	Merupakan rentang waktu yang masih dapat diterima oleh pengguna dan pengguna merasa masih belum ada interupsi
1 sampai 2 detik	Pengguna merasa adanya interupsi, namun masih dapat diterima
2 sampai 6 detik	Pengguna mulai merasa tidak senang dan tidak puas, namun masih dapat diterima
6 sampai 10 detik	Rentang waktu maksimal yang dapat diterima pengguna. Namun sebagian pengguna akan meninggalkan sistem saat memasuki rentang waktu ini.

IV. HASIL DAN PEMBAHASAN

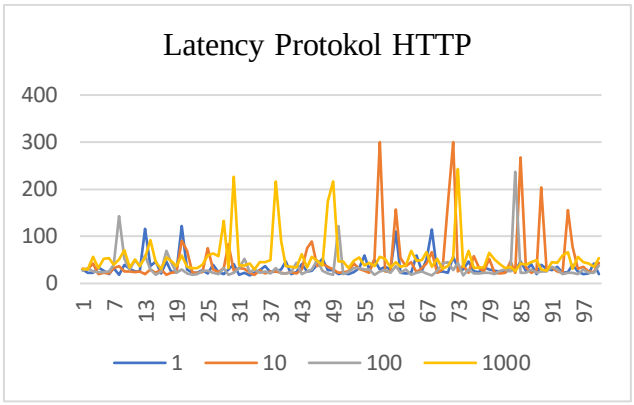
Pengujian dilakukan dengan menjalankan purwarupa kendaraan di dalam ruangan sebanyak parameter yang terdapat pada tabel I. Purwarupa tersebut akan berjalan mengelilingi ruangan dengan jalur yang acak. Setiap interval 1 detik, purwarupa tersebut akan mengirimkan data ke *server* dengan parameter yang telah ditentukan sebelumnya. Aktivitas purwarupa tersebut akan dihentikan ketika telah terdapat 100 data yang berhasil dikirimkan ke *server* atau dalam jangka waktu kurang lebih 100 detik setelah dinyalakan.

Berdasarkan hasil pengujian protokol *Web Socket*, apabila ditinjau dari ukuran data yang dikirimkan, pada protokol *Web Socket* tidak terjadi peningkatan yang signifikan seperti yang ditunjukkan pada Gambar 5. Saat data yang dikirimkan berukuran 1 sampai 100 karakter, nilai *latency* yang dihasilkan berada antara 10,75 sampai 13,06 milidetik. Namun ketika ukuran data yang dikirimkan berjumlah 1000 karakter, maka terjadi peningkatan rata-rata nilai *latency* yang cukup signifikan yaitu menjadi 23,29 milidetik. Meskipun demikian, nilai tersebut masih berada jauh di bawah ambang batas waktu respon terbaik berdasrakan Tabel II yaitu 100 milidetik, sehingga dapat disimpulkan bahwa nilai tersebut tidak berdampak apapun terhadap performa.



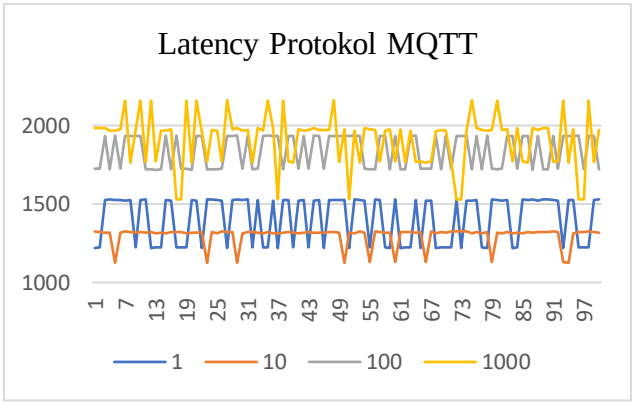
Gambar. 5 Relasi nilai *latency* protokol *web socket* terhadap ukuran data

Pada pengujian protokol HTTP, tidak terdapat perbedaan nilai *latency* untuk pengiriman data berukuran 1 hingga 1000 karakter seperti yang ditunjukkan pada Gambar 6. Nilai *latency* rata-rata berada pada angka 32,29 sampai dengan 61,19 milidetik, di mana angka 61,19 merupakan nilai rata-rata untuk data berukuran 10 karakter. Sehingga dapat disimpulkan bahwa besar kecilnya ukuran data yang dikirimkan (1-1000 karakter) tidak mempengaruhi *latency* dari protokol HTTP dalam mengirimkan data.



Gambar. 6 Relasi nilai *latency* protokol HTTP terhadap ukuran data

Pada pengujian protokol MQTT, terdapat pola peningkatan *latency* terhadap ukuran data yang dikirimkan seperti yang ditunjukkan pada Gambar 7. Pada data berukuran 1 dan 10 karakter, nilai rata-rata *latency* berada pada rentang 1300,28 sampai dengan 1407,26 milidetik. Sedangkan pada ukuran data 100 hingga 1000 karakter, nilai *latency* berada pada rentang 1836,29 sampai dengan 1909,12 milidetik. Hal ini menunjukkan bahwa ukuran data membawa dampak yang cukup signifikan terhadap *latency* pada protokol MQTT.



Gambar. 7 Relasi nilai *latency* protokol MQTT terhadap ukuran data

Berdasarkan keseluruhan pengujian yang telah dilakukan, didapatkan hasil seperti yang ditunjukkan pada Tabel III. Pada tabel tersebut dapat terlihat bahwa protokol *Web Socket* memiliki rata-rata waktu *latency* yang paling rendah di antara protokol yang lain, yaitu antara 10,75 sampai dengan 23,29 milidetik. Urutan berikutnya diikuti oleh protokol HTTP dengan nilai rata-rata *latency* antara 32,29 sampai ddengan 61,19 milidetik. Pada urutan terakhir, terdapat protokol MQTT dengan nilai rata-rata *latency* antara 1300,28 sampai dengan 1909,12 milidetik.

TABEL III
HASIL PENGUJIAN PENGIRIMAN DATA

Ukuran (byte)	Rata-Rata Waktu Transmisi Data (milidetik)		
	Web Socket	**HTTP**	**MQTT**
1	13,06	33,47	1407,26
10	11,18	61,19	1300,28
100	10,75	32,29	1836,29
1000	23,29	54,66	1909,12

Pada hasil pengujian tersebut, protokol *Web Socket* dan *HTTP* memiliki tingkat *latency* yang relatif lebih rendah apabila dibandingkan dengan protokol MQTT. Berdasarkan referensi pada Tabel II, keduanya masuk ke kategori terbaik karena nilai rata-rata *latency* keduanya kurang dari 0,1 detik (100 milidetik). Sementara itu, pada protokol MQTT nilai *latency* berada pada kategori 1 sampai 2 detik (1000 sampai 2000 milidetik), di mana rentang waktu tersebut masih dapat diterima meskipun pengguna merasa terdapat adanya interupsi pada sistem. *Latency* pada protokol MQTT memiliki nilai yang cukup tinggi karena data yang dikirimkan dari perangkat IoT harus dikirimkan kepada *server broker* terlebih dahulu untuk kemudian diteruskan pada pihak-pihak yang melakukan *subscribe* terhadap perangkat tersebut, termasuk *server* yang digunakan untuk penyimpanan data. Berbeda dengan protokol *Web Socket* dan HTTP yang dapat menghubungkan langsung perangkat IoT dengan *server*, sehingga nilai *latency* yang dihasilkan terbilang cukup rendah. Meskipun demikian, karena mekanisme *publisher* dan *subscriber* yang dimiliki oleh MQTT, maka banyaknya jumlah perangkat yang terhubung kepada *server broker* tidak berdampak banyak pada *latency* saat *publisher* mengirimkan data kepada *subscriber*. Berbeda dengan protokol *Web Socket* dan HTTP yang mengirimkan data secara berurutan (*sequential*), apabila terdapat 100 perangkat yang terhubung, maka akan terdapat 100 *request* sehingga diperlukan waktu sebanyak 100 dikalikan dengan nilai rata-rata *latency*. Berbeda dengan protokol MQTT, di mana ketika terdapat 100 perangkat yang terhubung, maka *publisher* hanya perlu mengirimkan data sebanyak satu kali saja, karena yang bertugas untuk meneruskan data tersebut adalah *server broker*.

Berdasarkan pengujian keseluruhan dengan pengukuran *latency*, maka dapat disimpulkan bahwa protokol terbaik adalah *Web Socket*. Namun pada dasarnya, *latency* bukan merupakan nilai mutlak untuk menentukan baik atau tidaknya sebuah protokol. Setiap protokol tetap memiliki karakteristik tersendiri yang tidak dapat dibandingkan satu sama lain. Misalnya pada protokol MQTT yang menggunakan mekanisme *publisher* dan *subscriber*, maka protokol ini sangat sesuai untuk implementasi dengan banyak perangkat yang terhubung dan membutuhkan data dari banyak sumber. Pada protokol *Web Socket*, karena nilai *latency* yang dimiliki sangat rendah, maka sangat cocok untuk implementasi sistem yang membutuhkan transmisi data secara *real-time* dan menuntut tingkat keamanan (*safety*) yang tinggi, seperti sistem untuk kendaraan. Sedangkan protokol HTTP memiliki fleksibilitas yang paling tinggi dibandingkan dengan protokol lain, karena protokol ini paling banyak memiliki dukungan pustaka maupun layanan pihak ketiga dibandingkan dengan *Web Socket* dan MQTT, sehingga HTTP merupakan protokol yang paling banyak diimplementasikan untuk berbagai macam keperluan pembangunan aplikasi seperti *web* dan *mobile*.

V. KESIMPULAN

Berdasarkan hasil pengujian *latency* dan evaluasi yang telah dilakukan, dapat disimpulkan bahwa protokol *Web Socket* dapat digunakan sebagai alternatif pengganti protokol MQTT yang selama ini paling banyak digunakan pada perangkat IoT. Kesimpulan tersebut didukung berdasarkan nilai rata-rata *latency* protokol *Web Socket* yang paling rendah dibanding protokol lain, yaitu 10,75 sampai 23,29 milidetik. Sehingga *Web Socket* merupakan protokol yang paling sesuai jika digunakan untuk membangun sistem yang bersifat *critical* atau berbasis *real-time* seperti pada kendaraan karena waktu jeda (*delay*) yang dihasilkan paling mendekati nilai nol. Meskipun demikian, nilai rata-rata *latency* seluruh protokol tetap berada pada ambang batas (di bawah 2 detik) yang masih dapat diterima oleh pengguna, sehingga seluruh protokol tetap layak untuk digunakan selamat sistem yang dibangun tidak bersifat *critical*. Selain itu pada protokol MQTT tidak memiliki mekanisme untuk mengirimkan umpan balik berupa *response* seperti protokol lainnya, sehingga apabila sistem membutuhkan kepastian bahwa data dapat tersampaikan dengan baik, maka hal ini tidak dapat dilakukan pada MQTT. Di sisi lain protokol MQTT lebih sesuai apabila digunakan untuk membangun sistem IoT dengan banyak *client* dalam wujud sensor, karena *latency* pada MQTT tidak dipengaruhi oleh jumlah *client* (*subscriber*) seperti pada protokol lain.

UCAPAN TERIMA KASIH

Terima kasih diucapkan kepada Fakultas Teknologi Informasi Universitas Kristen Duta Wacana yang telah mendukung kegiatan penelitian ini dalam bentuk berupa pendanaan penelitian dengan nomor kontrak 186/D.01/LPPM/2023.

REFERENSI

[1] K. A. Nugraha, "Deteksi Area Parkir Mobil Berbasis Marker Menggunakan Moment Invariants dan K-NN," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 5, no. 1, pp. 112-121, 2019.

[2] R. M. Yozienanda, W. Widiarto and A. Wijayanto, "Otomasi dan Monitoring Hidroponik pada Tanaman Selada dengan Metode Sonic Bloom Berbasis IoT," *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, vol. 8, no. 3, pp. 422-431, 2022.

[3] F. K. M. Rashid, O. S. Osman, E. T. Mcgee and H. Raad, "Discovering Hazards in IoT Architectures: A Safety Analysis Approach for Medical Use Cases," *IEEE Access*, vol. 11, pp. 53671 - 53686, 2023.

[4] W. Lim, S. Lee, J. Yang, M. Sunwoo, Y. Na and K. Jo, "Automatic Weight Determination in Model Predictive Control for Personalized Car-Following Control," *IEEE Access*, vol. 10, pp. 19812 - 19824, 7 Februari 2022.

[5] A. Biswas and H.-C. Wang, "Autonomous Vehicles Enabled by the Integration of IoT, Edge Intelligence, 5G, and Blockchain," *Sensors,* vol. 23, no. 4, p. 2023, 2023.

[6] S. Bagwari, A. Roy, R. Singh and A. Gehlot, "Disaster Monitoring based on IoT and Long Range Assisted Framework," *Journal of Physics: Conference Series,,* vol. 2327, no. 1, pp. 12-20, 2022.

[7] Supriadi, A. Wajiansyah and A. B. W. Putra, "Prototipe Peringatan Dini Banjir dengan Menerapkan Teknologi Internet of Thing," *Jurnal Edukasi dan Penelitian Informatika,* vol. 7, no. 1, pp. 31-38, 2021.

[8] U. Ristian, I. Ruslianto, H. Hasfani and K. Sari, "Perancangan Arsitektur Node Nirkabel dalam Efisiensi Bandwidth Smart Greenhouse Berbasis Protokol MQTT," *Jurnal Edukasi dan Penelitian Informatika,* vol. 9, no. 2, pp. 218-225, 2023.

[9] M. U. Saleem, M. R. Usman, M. A. Usman and C. Politis, "Design, Deployment and Performance Evaluation of an IoT Based Smart Energy Management System for Demand Side Management in Smart Grid," *IEEE Access,* pp. 15261 - 15278, 2022.

[10] U. Khalil, Mueen-Uddin, O. A. Malik and S. Hussain, "A Blockchain Footprint for Authentication of IoT-Enabled Smart Devices in Smart Cities: State-of-the-Art Advancements, Challenges and Future Research Directions," *IEEE Access,* pp. 76805 - 76823, 2022.

[11] G. Mani, J. K. Viswanadhapalli and P. Sriramalakshmi, "AI powered IoT based Real-Time Air Pollution Monitoring and Forecasting," *Journal of Physics: Conference Series,* vol. 2115, no. 1, pp. 12-16, 2021.

[12] S. Lee, H. Choi, T. Kim, H.-S. Park and J. K. Choi, "A Novel Energy-Conscious Access Point (eAP) System With Cross-Layer Design in Wi-Fi Networks for Reliable IoT Services," *IEEE Access,* vol. 10, pp. 61228 - 61248, 2022.

[13] H. Gao, X. Qin, R. J. D. Barroso, W. Hussain, Y. Xu and Y. Yin, "Collaborative Learning-Based Industrial IoT API Recommendation for Software-Defined Devices: The Implicit Knowledge Discovery Perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 6, no. 1, pp. 66 - 76, 2022.

[14] P. A. M. Serrano and J. J. S. Oñate, " Integration of RESTFul API to Student Information System for Secured Data Sharing and Single Sign-on," in *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, Manila, Philippines, 2021.

[15] A. Agocs and J.-M. L. Goff, "A web service based on RESTful API and JSON Schema/JSON Meta Schema to construct knowledge graphs," in *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Alsace, Colmar, France, 2018.

[16] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," *IEEE Access ,* vol. 8, pp. 201071 - 201086, 2020.

[17] F. Buccafurri, V. d. Angelis and S. Lazzaro, "MQTT-A: A Broker-Bridging P2P Architecture to Achieve Anonymity in MQTT," *IEEE Internet of Things Journal,* vol. 10, no. 17, pp. 15443 - 15463, 2023.

[18] C.-S. Park and H.-M. Nam, "Security Architecture and Protocols for Secure MQTT-SN," *IEEE Access,* vol. 8, pp. 226422 - 226436, 2020.

[19] A. M. Zambrano, M. Zambrano, E. L. O. Mejía and X. Calderón, "SIGPRO: A Real-Time Progressive Notification System Using MQTT Bridges and Topic Hierarchy for Rapid Location of Missing Persons," *IEEE Access,* vol. 8, pp. 149190 - 149198, 2020.

[20] A. Eduardo, A. Michael and C. Miguel, "Optimization and improvement of bidirectional connections with Web Socket on Synapse framework using Web Workers technology," in *2021 IEEE Sciences and Humanities International Research Conference (SHIRCON)*, Lima, Peru, 2021.

[21] S. S. K. Jaeyoung Hwang Sejong University, L. Nkenyereye, N. Sung, J. Kim and J. Song, "IoT Service Slicing and Task Offloading for Edge Computing," *IEEE Internet of Things Journal* , pp. 11526 - 11547, 2021.

[22] G. Y. Odongo, R. Musabe and D. Hanyurwim, "An Efficient LoRa-Enabled Smart Fault Detection and Monitoring Platform for the Power Distribution System Using Self-Powered IoT Devices," *IEEE Access,* pp. 73403 - 73420, 2022.

[23] Syaifurrahman and A. Aula, "Sistem Monitoring dan Proteksi pada Stop Kontak Berbasis IoT," *Jurnal Edukasi dan Penelitian Informatika,* vol. 8, no. 1, pp. 104-110, 2022.

[24] T. Widodo, A. B. Santoso, S. I. Ishak and R. Rumeon, "Journal of Physics: Conference Series," *Jurnal Edukasi dan Penelitian Informatika,* vol. 9, no. 1, pp. 59-66, 2023.

[25] I. Ruslianto, U. Ristian and H. Hasfani, "Sistem Pintar Untuk Anggur (Sipunggur) pada Kawasan Tropis Berbasis Internet of Things (IoT)," *Jurnal Edukasi dan Penelitian Informatika,* vol. 8, no. 1, pp. 121-127, 2022.

[26] S. N. Swamy and S. R. Kota, "An Empirical Study on System Level Aspects of Internet of Things (IoT)," *IEEE Access* , pp. 188082 - 188134, 2020.

[27] E. Staniloiu, R. Nitu, R. Aron and R. Rughinis, "Extending Client-Server API Support for Memory Safe Programming Languages," in *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Iasi, Romania, 2021.

[28] S. Routray, D. Raipure, S. R. N. K and N. P. Shetty, "Secure Sharing of Text Based Data Using Hybrid Encryption Algorithms in a Client-Server Model," in *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Erode, India, 2021.

[29] Z. Lei, Y. Chen, Y. Yang, M. Xia and Z. Qi, "Bootstrapping Automated Testing for RESTful Web Services," *IEEE Transactions on Software Engineering,* vol. 49, no. 4, pp. 1561 - 1579, 2022.

[30] S.-P. Ma, Y.-J. Chen, Y. Syu, H.-J. Lin and Y.-Y. FanJiang, "Test-Oriented RESTful Service Discovery with Semantic Interface Compatibility," *IEEE Transactions on Services Computing,* vol. 14, no. 5, pp. 1571 - 1584, 2021.

[31] Q. Liu and X. Sun, "Research of Web Real-Time Communication Based on Web Socket," *International Journal of Communications, Network and System Sciences,* vol. 5, no. 12, pp. 797-801, 2012.

[32] K. A. Nugraha, "Real-Time Bus Arrival Time Estimation API using WebSocket in Microservices Architecture," *International Journal on Advanced Science, Engineering and Information Technology (IJASEIT),* vol. 13, no. 3, pp. 1018-1024, 2023.

[33] T. Hamilton, "Response Time Testing – How to Measure for API?," Guru99, 14 Oktober 2023. [Online]. Available: https://www.guru99.com/response-time-testing.html#:~:text=Generally%2C%20response%20time%20should%20be,lesser%20costs%2C%20higher%20customer%20satisfaction.. [Accessed 16 Oktober 2023].