Implementasi dan Analisis Protokol Komunikasi IoT untuk *Crowdsensing* pada Bidang Kesehatan

Ata Amrullah[1], M. Udin Harun Al Rasyid[2*], Idris Winarno[3]
Politeknik Elektronika Negeri Surabaya (PENS)
Jl. Raya ITS, Kampus PENS Sukolilo, Surabaya, Indonesia
Email: ataislucky@gmail.com[1], udinharun@pens.ac.id[2], idris@pens.ac.id[3]
Corresponding author[*]: M. Udin Harun Al Rasyid

Abstract - The development of information and communication technology has marked the era of the fourth industrial revolution. The ease of data exchange between mobile devices creates a new paradigm in centralized data collection called crowdsensing. In the health sector, crowdsensing no longer relies on cellphones as a means of gathering information due to the limitations of sensors embedded in cellphones. Various studies using crowdsensing rely on the capabilities of Internet of Things (IoT) devices. In the health sector, crowdsensing can help get a lot of important data about the general health condition of the community. However, the existing crowdsensing technique uses only one communication protocol. This method can cause problems if the IoT device uses various communication protocols. Therefore, we propose a multi-protocol communication gateway architecture for crowdsensing that acts as a sensor data receiver with various communication protocols. We tried to combine the three types of communication protocols that run on the gateway, namely MQTT, HTTP, and CoAP. This gateway also converts all three protocols to the cloud's back-end server's protocol. The test results show that the gateway is able to get data well even though the three protocols are run simultaneously. CoAP protocol has better performance than both protocols in throughput measurement. The MQTT protocol has the best performance in delay measurement.

Keywords – Crowdsensing, Internet of Things, Multi-Communication Protocol, Interoperability.

Intisari - Perkembangan teknologi informasi dan komunikasi telah menandai berlangsungnya era revolusi industri 4.0. Kemudahan pertukaran data antar perangkat yang bergerak menjadikan paradigma baru pada pengumpulan data terpusat yang disebut *crowdsensing*. Pada bidang kesehatan, *crowdsensing* tidak lagi mengandalkan telepon bergerak sebagai perangkat pengumpul informasi karena keterbatasan sensor tertanam pada telepon. Berbagai penelitian menggunakan *crowdsensing* telah mengandalkan kemampuan dari perangkat Internet of Things (IoT). *Crowdsensing* pada sektor kesehatan dapat membantu mengumpulkan sumber data yang substansial tentang kondisi kesehatan masyarakat secara umum. Namun, kebanyakan teknik *crowdsensing* hanya mengandalkan satu protokol komunikasi. Metode ini dapat menyebabkan masalah jika perangkat IoT menggunakan protokol komunikasi yang beragam jenisnya. Oleh karena itu, kami mengusulkan arsitektur gateway multi-protokol komunikasi untuk crowdsensing yang berperan sebagai penerima data sensor yang beragam protokol komunikasinya. Kami mencoba menggabungkan ketiga jenis protokol komunikasi yang dijalankan pada gateway yaitu MQTT, HTTP dan CoAP. Gateway ini juga berfungsi untuk mengubah ketiga protokol ke dalam protokol yang sama dengan back-end server di cloud. Hasil pengujian menunjukkan bahwa gateway mampu menerima data dengan baik meskipun ketiga protokol dijalankan secara bersamaan. Protokol CoAP memiliki kinerja yang lebih baik daripada kedua protokol dalam pengujian *throughput*. Protokol MQTT memiliki performa terbaik pada pengukuran *delay*.

Kata Kunci – *Crowdsensing, Internet of Things*, Protokol Multi-Komunikasi, *Interoperability*.

I. PENDAHULUAN

Pada dekade terakhir adalah mulainya era revolusi industri 4.0. Pemanfaatan teknologi dan komunikasi merupakan pemantik dari revolusi industri ini. Perangkat berukuran mikro bisa melakukan interkoneksi dengan beberapa perangkat maupun platform. Salah satu pilar dari industri keempat adalah *Internet of Things* (IoT) [1]. Beberapa perangkat IoT telah berkembang dari bersifat statis menjadi portabel. Kelebihan inilah yang kemudian memberikan dampak pada penggunaan perangkat IoT untuk *crowdsensing*.

Crowdsensing didefinisikan sebagai teknik pengumpulan data oleh sekelompok besar pengguna untuk mendapatkan informasi dengan tujuan yang sama dan menggunakan perangkat bergerak [2], [3]. Paradigma *crowdsensing* terus mendapat perhatian dan penelitian yang meningkat tentang teknik ini. Beberapa penelitian telah menggunakan teknik *crowdsensing* pada bidang sosial, infrastruktur, lingkungan, dan kesehatan [4]–[6]. Pada bidang kesehatan, *crowdsensing* dapat memberikan sumber data yang substansial untuk dapat lebih memahami kondisi kesehatan seseorang secara individual maupun dalam komunitas.

Namun, perangkat IoT untuk *crowdsensing* akan mendapat masalah interoperabilitas jika pusat pengumpulan data hanya menerima satu protokol komunikasi. Perangkat *crowdsensing* yang memiliki domain berbeda seperti API (Application Programming Interface) dan protokol komunikasi akan kesulitan berkomunikasi dengan perangkat lain yang juga mengirimkan data ke server cloud. Hal ini akan menjadi kendala dalam proses komunikasi karena perangkat yang digunakan dapat berada pada protokol domain yang berbeda. Oleh karena itu, kami mengusulkan arsitektur lokal gateway multi-protokol pada teknik *crowdsensing* menggunakan beberapa protokol komunikasi seperti HTTP (Hypertext Transfer Protocol), MQTT (Message Queuing Telemetry Transport) dan CoAP (Constrained Application Protocol) dalam satu gateway tersebut sebelum mengirimkan data ke platform IoT di server cloud.

Makalah ini disusun sebagai berikut: bagian 2 menyajikan penelitian terkait, indikator kinerja sistem serta arsitektur sistem yang diusulkan. Bagian 3 menyajikan hasil dan pembahasan dari uji kinerja ketiga protokol yang diusulkan. Hasil kesimpulan dirangkum pada bagian 4.

II. SIGNIFIKANSI STUDI

Berikut ini menjelaskan signifikan studi dari penelitian ini yang terdiri dari studi literatur terkait atau penelitian terdahulu dan sistem desain.

A. Studi Literatur

Penelitian Jovanović dkk. [7] mengusulkan implementasi pengumpulan data dari pasien hipertensi untuk dijadikan sebagai database eksperimental menggunakan Platform as a Service (PaaS) di cloud. Beberapa tantangan yang dihadapi oleh mereka adalah analisis data tanpa melibatkan fitur tekanan darah, dan juga keandalan data, bandwidth terbatas serta sumber daya baterai. Mereka menggunakan sensor tambahan, yaitu ECG sensing extender untuk mengubah sinyal biomedis menjadi audio, sehingga memudahkan pemrosesan di smartphone. Seharusnya perangkat tambahan sudah dapat terintegrasi dengan back-end server sehingga pelaku *crowdsensing* dapat melakukan penginderaan dengan mudah.

Penelitian Marjanovic dkk. [8] mengusulkan sebuah arsitektur Mobile Edge Computing (MEC). MEC memberikan solusi pada teknik *crowdsensing* dengan memindahkan komputasi ke edge, karena arsitektur berbasis MEC memungkinkan peningkatan kinerja yang signifikan karena pembagian ruang masalah berdasarkan lokasi, di mana pemrosesan dan agregasi data waktu nyata dilakukan dekat dengan sumber data. Arsitektur ini mampu mengurangi lalu lintas terkait di inti seluler dan akan memfasilitasi penerapan *crowdsensing* dalam skala besar. Selain peningkatan kinerja, arsitektur yang diusulkan mengurangi ancaman privasi dan mengizinkan pengguna untuk mengontrol aliran data sensor yang disumbangkan. Penelitian

ini disebutkan mampu mengurangi lalu lintas data pada seluler. Tetapi, akan kesulitan mendapatkan data sensor yang berbeda protokol komunikasi.

Penelitian Pryss dkk. [9] mengusulkan penggunaan API untuk memenuhi kebutuhan peneliti dari domain medis. Khususnya, API harus mampu menangani perubahan persyaratan secara fleksibel. Penelitian yang dikerjakan oleh mereka menyajikan kemudahan evaluasi data dalam konteks manajemen gangguan kronis. Penelitian ini masih memerlukan pengembangan lebih lanjut karena implementasi hanya dilakukan untuk penyakit tinnitus.

Penelitian Zambrano dkk [10] mengembangkan sistem komunikasi real-time sebagai upaya pencegahan dan pencarian korban penculikan. Tujuan penelitian mereka adalah menawarkan solusi inovasi teknologi berbasis cloud secara real-time yang memanfaatkan komunitas (sekumpulan orang) menggunakan smartphone dengan didukung oleh protokol komunikasi IoT, yaitu MQTT sehingga mampu untuk mengurangi waktu penyelamatan. Pada penelitian ini, implementasi hanya dilakukan menggunakan aplikasi pada smartphone tanpa ada integrasi dengan back-end server.

Maqbool dkk [11] mengusulkan sistem pemantauan pasien waktu nyata berbasis IoT yang menggunakan Message Queuing Telemetry Transport (MQTT) untuk pengiriman sinyal Elektrokardiogram (EKG) dari aplikasi yang diusulkan ke server web. Mereka mengandalkan protokol komunikasi tersebut untuk mengurangi beberapa masalah pelayanan kesehatan profesional seperti kekurangan dokter, rumah sakit, dan jarak yang jauh dari kota. Mereka menitikberatkan penelitian mereka pada jumlah atau kuantitas data yang diterima dari perangkat EKG yang digunakan. Hasil analisis penelitian mereka telah menunjukkan bahwa tidak ada kehilangan paket data dan kesalahan paket baik pada lingkungan jaringan LAN maupun jaringan WAN.

Berdasarkan kajian pustaka di atas, maka dapat dideskripsikan beberapa perbandingan dalam bentuk table I dibawah ini:

TABEL I
PERBANDINGAN PENELITIAN TERKAIT

No.	Ref.	Mekanisme	Perangkat	Partisipan	Protokol	Solusi
1	[7]	Database eksperimental menggunakan Layanan Cloud	ECG sensing extender	10 pengguna	Seluler	Pengumpulan data dan motivasi user
2	[8]	Arsitektur Mobile Edge Computing (MEC)	Simulasi	85 pengguna (simulasi)	Seluler	Interoperabilitas
3	[9]	Mendesain API untuk m-Health	Smart phone	-	HTTP/ RESTful API	Manajemen data dan interoperabilitas
4	[10]	Menggunakan aplikasi admin dan partisipan	Smart phone	50 pengguna	MQTT	Pengurang-an waktu
5	[11]	Mengirim sinyal EKG dari aplikasi menuju web server	NodeMCU, sensor ECG	-	MQTT	Pemantauan kesehatan jarak jauh
6	Sistem yang diusulkan		Raspberry Pi, NodeMCU, sensor mlx, sensor ecg	5 pengguna dan ±100 skrip simulasi sensor	MQTT, HTTP, CoAP, WebSocket	Pengumpulan data, Manajemen data, dan interoperabilitas

B. Key Performance Indicator

Untuk mengevaluasi kinerja protokol IoT yang dipilih selama eksperimen *crowdsensing*, *key performance indicator* (KPI) harus ditentukan. KPI merupakan cara untuk mengetahui kualitas dari suatu layanan pada jaringan tertentu. Pengukuran kualitas tersebut dilakukan berdasarkan standar dari Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) [12] serta International Telecommunication Union-Telecommunication (ITU-T) [13]. Dalam penelitian ini parameter yang akan dianalisis adalah *throughput*, *delay*, dan *packet loss*.

1. Throughput

Throughput diartikan sebagai laju data aktual per satuan waktu (bits per second). Biasanya *throughput* selalu dikaitkan dengan bandwidth karena *throughput* memang bisa disebut sebagai bandwidth dalam kondisi yang sebenarnya. Bandwidth lebih bersifat tetap sementara *throughput* sifatnya dinamis tergantung trafik yang sedang terjadi.

Untuk menghitung *throughput* digunakan rumus:

$$Throughput = \frac{Paket\ data\ yang\ diterima}{Lama\ pengamatan} \quad (1)$$

2. Delay

Delay didefinisikan sebagai selisih waktu pengiriman sebuah paket saat dikirimkan dengan saat paket tersebut diterima pada node tujuan. *Delay* disebut juga dengan istilah *latency* terdiri dari beberapa faktor penundaan yaitu propagation *delay* atau transmission *delay* yaitu penundaan akibat waktu tempuh paket selama dalam saluran transmisi yang bandwidth-nya berbeda-beda, queuing *delay* yaitu waktu antrian paket sebelum dilewatkan pada saluran transmisi dan lainnya. Waktu tunda dinyatakan dalam satuan detik.

Menurut versi TIPHON, standardisasi nilai *delay* adalah sebagai berikut:

TABEL II
KATEGORI DELAY VERSI TIPHON

Kategori Delay	Besar Delay	Indeks
Sangat Baik	<150 ms	4
Baik	150 s/d 300 ms	3
Cukup	300 s/d 450 ms	2
Kurang Baik	>450 ms	1

Sedangkan menurut versi ITU-T, standardisasi *delay* sebagai berikut :

TABEL III
STANDAR DELAY VERSI ITU-T

Nilai Delay	Kualitas
< 150 ms	Baik
150 s/d 400 ms	Cukup
> 400 ms	Buruk

Untuk menghitung rata-rata *delay* digunakan rumus:

$$Delay\ rata - rata = \frac{Total\ delay}{Total\ paket\ yang\ diterima} \quad (2)$$

3. Packet Loss

Pengukuran *packet loss* sebagai bahan analisis jaringan pada komunikasi data secara real time cukup penting. Trafik komunikasi real time yang menggunakan transport protokol UDP tidak dapat menjamin sebuah paket data dapat diterima oleh node tujuan dengan baik. Berbeda dengan pengiriman paket data menggunakan protokol TCP yang proses pengiriman datanya melalui proses three-way-handshaking. Dengan demikian, perlu dipastikan kualitas dari protokol komunikasi IoT yang digunakan secara real time.

Untuk menghitung *packet loss* digunakan rumus sebagai berikut:

$$Packet\ loss\ rate = \frac{(total\ paket\ hilang)}{(total\ paket\ yang\ dikirim)} x100\% \quad (3)$$

Adapun standardisasi nilai persentase *packet loss* versi TIPHON adalah sebagai berikut:

TABEL IV
STANDAR PACKET LOSS VERSI TIPHON

Kategori Delay	Packet Loss	Indeks
Sangat Baik	0 %	4
Baik	3 %	3
Cukup	15 %	2
Kurang Baik	>25 %	1

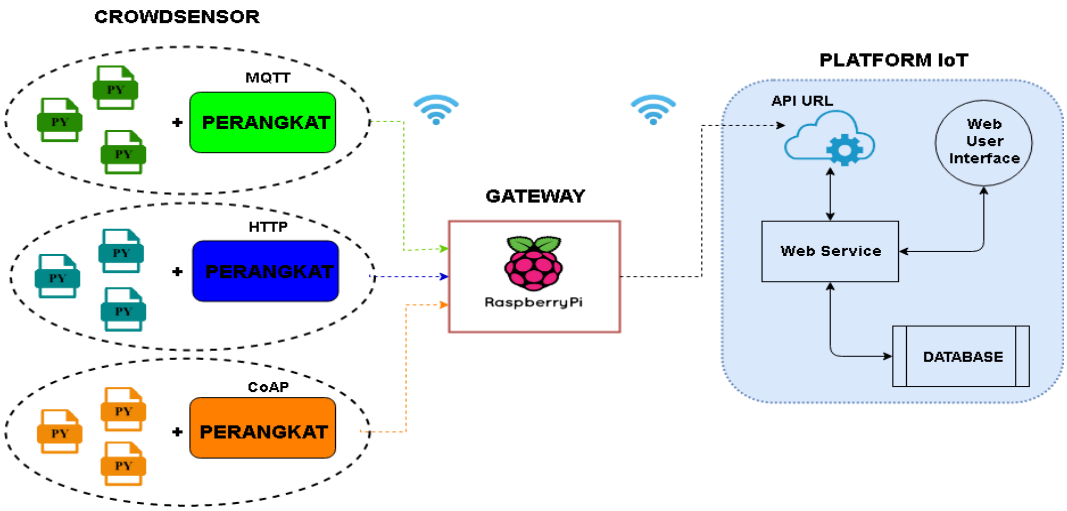
Sedangkan menurut versi ITU-T, terdapat tiga kategori penurunan kualitas jaringan berdasarkan standardisasi nilai *packet loss* sebagai berikut

TABEL V
STANDAR PACKET LOSS VERSI ITU-T

Packet Loss	Kualitas
3%	Baik
15%	Cukup
>25%	Buruk

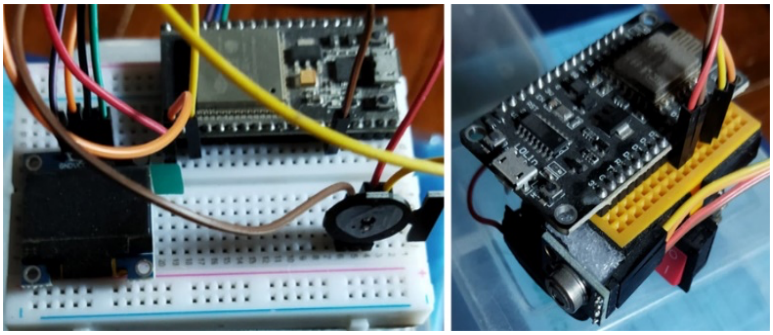
C. Arsitektur Sistem

Bagian ini menjelaskan arsitektur sistem yang digunakan pada penelitian ini. Gambar 1 menjelaskan arsitektur sistem yang terdiri dari komponen node sensor, gateway dan platform IoT yang ada di cloud.



Gambar 1. Arsitektur sistem

Pada penelitian ini, node sensor merupakan perangkat portabel yang terdiri dari sensor kesehatan, mikrokontroler, baterai, dan oled display seperti terlihat pada Gambar 2. Untuk mengimplementasikan teknik *crowdsensing*, maka sebanyak lima perangkat portable didukung dengan 100 program Python (.py) yang bertindak sebagai simulator sensor pengirim pesan. Setiap perangkat menyimpan data sensor, email pengguna serta token rahasia untuk hak akses pengiriman pesan ke platform pada cloud. Selain itu, tiap perangkat juga telah dibagi ke dalam grup protokol komunikasi yang berbeda, yaitu HTTP, MQTT dan CoAP. Pertimbangan penggunaan ketiga protokol ini mengacu pada perkembangannya untuk digunakan pada sistem IoT secara luas [14].



Gambar 2. Perangkat node sensor

Pertimbangan penggunaan program Python sebagai simulator sensor pengirim pesan adalah perbandingan dengan real sensor yang tidak berbeda jauh, meliputi nilai *throughput* dan *delay*. Jika dikalkulasikan, nilai *throughput* dan *delay* antara perangkat real sensor dengan program Python hampir sama dengan nilai error yang sangat kecil. Nilai error pada *throughput* adalah kurang dari 1 bit/sec, sedangkan nilai error pada *delay* kurang dari 1 detik.

Selanjutnya, komponen kedua dari penelitian ini adalah multi-protokol gateway yaitu Raspberry pi tipe 4B yang dapat memproses data yang dikirim dari sensor baik menggunakan protokol MQTT, HTTP maupun CoAP kemudian dikirim ke cloud server menggunakan protokol Websocket. Adapun detail dari gateway akan dijelaskan pada Gambar 5.

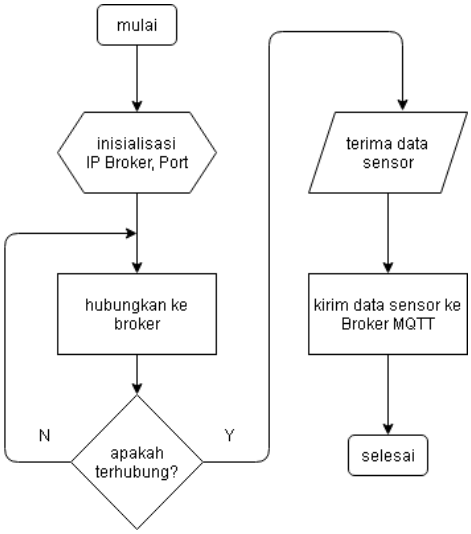
Komponen ketiga dari penelitian ini adalah cloud server, platform IoT kesehatan, yang berfungsi sebagai platform untuk pengumpulan data, pemrosesan, visualisasi, dan manajemen data pasien. Untuk mendaftarkan perangkat yang akan digunakan pada platform IoT kesehatan, perangkat tersebut harus ditambahkan ke akun yang berdaftar. Setelah perangkat ditambahkan, token akses dibuat. Token akses ini diperlukan agar platform tersebut dapat mengumpulkan data telemetri dari perangkat dengan protokol IoT. Hingga penelitian ini dibuat, platform tersebut masih hanya menerima komunikasi menggunakan Websocket/ HTTP.

1. *Pengiriman Data Sensor*

Pengiriman data sensor menggunakan ketiga protokol komunikasi.

a. *Pengiriman Data Sensor Menggunakan MQTT*

Setelah data sensor diperoleh, data tersebut akan dikirim ke broker MQTT seperti yang disajikan pada Gambar 3 di bawah ini:

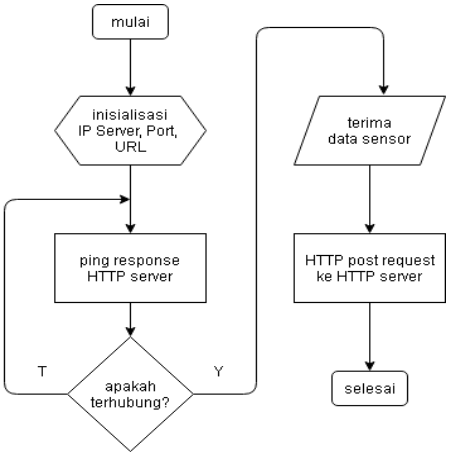


Gambar 3. Flowchart publisher MQTT

Pada gambar 3, sensor node menginisialisasi IP broker dan port yang ada pada Raspberry Pi. Selanjutnya, proses otentikasi koneksi MQTT Broker berdasarkan data yang diinisialisasi. Kemudian data dipublikasikan ke MQTT Broker. Memublikasikan data hanya sekali per node sensor. Hal ini dilakukan untuk menggambarkan metode *crowdsensing* beberapa node sensor pada satu waktu.

b. Pengiriman Data Sensor Menggunakan HTTP

Seperti pada publisher MQTT, setelah data sensor didapatkan, data tersebut juga dikirimkan ke server HTTP.

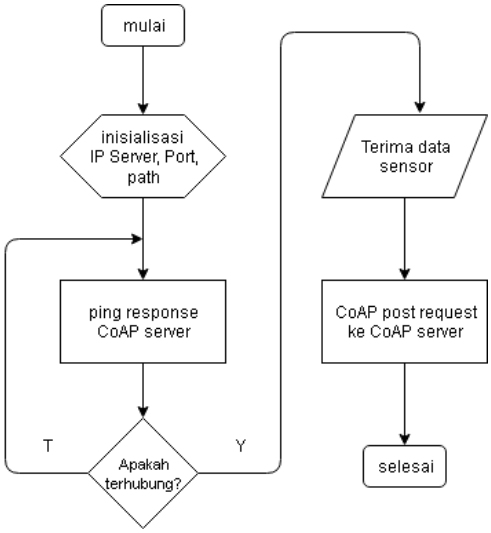


Gambar 4. Flowchart publisher HTTP

Pada gambar 4, node sensor menginisialisasi IP server HTTP dan port pada Raspberry Pi. Proses selanjutnya adalah otentikasi. Kemudian klien HTTP akan melakukan ping ke server HTTP. Jika Ping berhasil, data sensor dikirim dengan fungsi HTTP POST Request.

c. Pengiriman Data Sensor Menggunakan CoAP

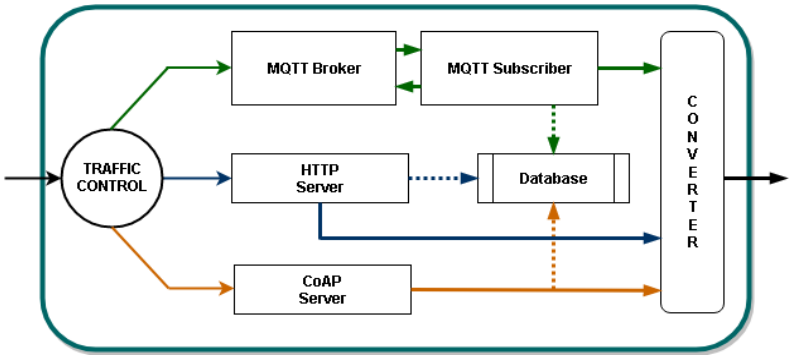
Pada gambar 5, setelah data sensor didapatkan, data tersebut dikirimkan ke server CoAP, namun hanya perlu satu kali ping response server. Jika sudah mendapat response, maka CoAP client akan mengirim data sensor tanpa memerlukan response dari server lagi.



Gambar 5. Flowchart publisher CoAP

2. *Multi-Communication Protocol Gateway*

Node gateway bertindak sebagai penghubung antara node sensor dan server cloud. Oleh karena itu, node gateway terdiri dari *traffic control*, broker MQTT, subscriber MQTT, server HTTP, server CoAP, serta database. Arsitektur gateway dapat dilihat pada Gambar 6.

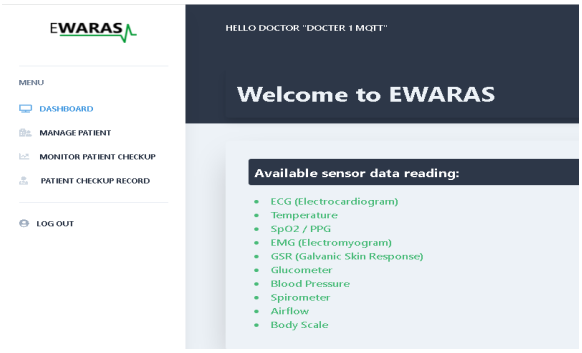


Gambar 6. Arsitektur *gateway* dari sensor ke platform IoT

Traffic Control berfungsi untuk mengontrol paket data yang masuk dengan cara memberi *delay* serta *reordering* paket data. Hal ini dilakukan untuk meminimalkan tabrakan data dari sensor menuju gateway yang bisa menyebabkan data hilang. Framework Mosquito [15] digunakan sebagai broker yang dijalankan pada Raspberry Pi. Kemudian, aplikasi subscriber MQTT pada Raspberry Pi sebagai node gateway adalah program Python dan menggunakan library paho.mqtt [16]. Supaya dapat digunakan sebagai subscriber MQTT, maka perlu diinstal library paho.mqtt terlebih dulu. Paho.mqtt mendukung berbagai perintah MQTT, termasuk publish dan subscribe. Adapun *converter* berfungsi mengubah koneksi ketiga protokol menjadi koneksi *websocket* supaya bisa melakukan transmisi data ke platform IoT.

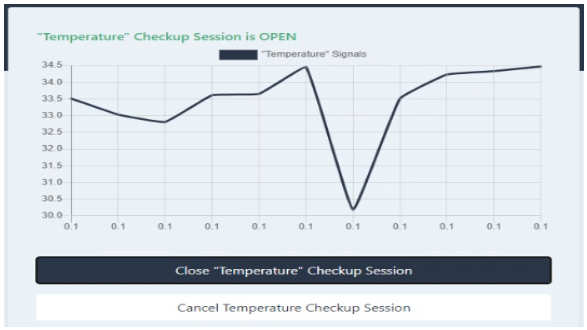
3. *Visualisasi pada Platform IoT*

Platform IoT kesehatan yang digunakan sebagai pusat pengumpulan data adalah sebuah platform berbasis web yang ada pada *cloud*. Platform tersebut menggunakan protokol websocket yang dilengkapi dengan API untuk autentikasi perangkat sensor yang digunakan.



Gambar 7. Halaman pengguna platform IoT

Adapun kategori pengguna pada platform ini adalah admin, dokter dan pasien. Pada gambar 7, kategori admin sama halnya dengan seorang operator yang memiliki akses untuk mendaftarkan dokter yang bekerja serta pasien yang berobat. Sehingga fungsi dari admin adalah mengontrol siapa saja yang bisa menggunakan platform dengan menu utama manage doctor, manage patient dan patient check-up record. Selanjutnya pada kategori patient, terdapat beberapa menu penerimaan data sensor yang bisa digunakan seperti ECG, Temperature, SPO2, EMG, GSR, Glucometer, Blood Pressure, Spirometer, Airflow dan Body Scale. Semua menu sensor dilengkapi dengan sub-menu monitor dan check-up record. Untuk melakukan check-up, maka seorang pasien perlu open ticket pada sub-menu monitor. Sedangkan pada pengguna dokter, terdapat tiga menu utama, yaitu manage patient, monitor patient check-up dan patient check-up record. Untuk pengukuran terhadap ketiga protokol, maka pengguna dokter dibagi menjadi tiga jenis, yaitu dokter mqtt, dokter http dan doket coap. Hal ini, dilakukan untuk memudahkan analisis terhadap keberhasilan transfer data menggunakan ketiga protokol kepada platform. Gambar 6 menunjukkan halaman pengguna dokter.



Gambar 8. Visualisasi penerimaan data sensor

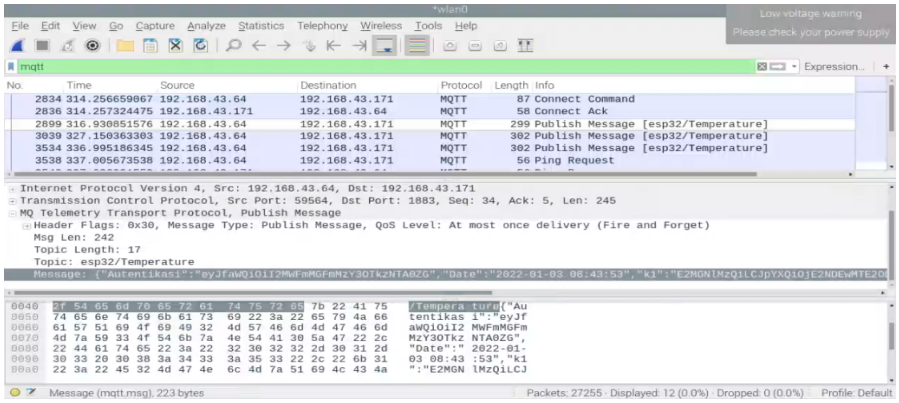
Setelah permintaan open ticket berhasil, selanjutnya data dari gateway akan diteruskan ke platform dengan visualisasi seperti pada gambar 8. Visualisasi tersebut adalah hasil pengamatan semua pengguna yang sedang melakukan pemantauan kesehatan dan setiap pengguna akan mendapatkan hasil pemantauan kesehatan setelah close check-up session.

III. HASIL DAN PEMBAHASAN

Pada penelitian ini dilakukan pengujian *delay*, pengujian *throughput*, dan pengujian kuantitas data pada gateway dan cloud server berdasarkan transfer data menggunakan protokol MQTT, HTTP dan CoAP. Penggunaan ketiga protokol komunikasi tersebut akan dievaluasi untuk *delay* pengiriman dan *throughput*. Jumlah waktu yang diperlukan untuk pengiriman untuk tiba di tujuan disebut sebagai penundaan. Saat mengunduh file, *throughput*

adalah bandwidth aktual yang diukur pada waktu tertentu menggunakan rute internet tertentu. Panjang paket (L, panjang paket (bit/s)) dibagi dengan bandwidth link (R, bandwidth link (bit/s)) untuk mendapatkan *delay*. Skenario *delay* dan pengujian *throughput* transmisi data pada ketiga protokol adalah pengujian *delay* dan *throughput* pada masing-masing protokol dari sensor node 5, 10, 15, 20, 30, dan 35 ke Raspberry Pi sebagai node gateway. Setiap sensor mengirim data sebanyak lima kali pengiriman. Ada dua opsi pengujian: yang pertama adalah uji tunda dan *throughput* pada pengiriman protokol MQTT, HTTP dan CoAP, yang kedua adalah uji tunda dan *throughput* pada pengiriman ketiga protokol secara bersamaan. Software Wireshark [17] pada Raspberry Pi digunakan untuk mengamati *delay* dan tes *throughput*. Selain itu, pengujian juga dilakukan untuk jumlah data pada gateway dan server cloud.

Penggunaan Wireshark untuk tracing ketiga protokol dengan cara menyaring sesuai nama protokol, seperti ditunjukkan pada gambar 9. Pada gambar tersebut terlihat, hanya protokol MQTT yang ditampilkan. Kolom Source berisi ip address 192.168.43.64, yang merupakan ip address dari perangkat *crowdsensing*. Sedangkan pada kolom Destination, 192.168.43.171 adalah ip address gateway (Raspberry pi).



Gambar 9. Pelacakan trafik protokol

A. Skenario I (satu protokol, satu set snsor)

Pengujian kinerja pertama dilakukan dengan skenario mengirimkan data sensor melalui satu protokol dari satu set node sensor. Misalnya, sepuluh node sensor MQTT mengirimkan data pada awalnya. Jadi, perangkat HTTP dan CoAP dimatikan. Prosedur ini dilakukan secara bergantian.

Tabel VI menunjukkan bahwa sistem hanya mengirimkan data melalui protokol MQTT, yang menunjukkan bahwa protokol MQTT ON dan protokol HTTP maupun CoAP OFF. Rata-rata pengukuran latency MQTT adalah 2,36 mili detik, sedangkan pengukuran *throughput* MQTT adalah 4,822 kbps. *Packet Loss* sebesar 1% terjadi ketika 35 perangkat secara bersamaan mengirim data sensor.

TABEL VI
PENGUJIAN DELAY DAN THROUGHPUT PADA MQTT SKENARIO I

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
5	9811	63.997	1.226	1.07	0%
10	17493	50.124	2.791	1.15	0%
15	26121	55.369	3.774	1.23	0%

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
20	34725	55.412	5.013	1.60	0%
25	43375	53.156	6.527	2.12	0%
30	51957	59.697	6.962	2.87	0%
35	60874	65.261	7.462	6.52	1%
AVERAGE			**4.822**	**2.36**	**0%**

Tabel VII menunjukkan bahwa sistem hanya mengirimkan data melalui protokol HTTP, yang menunjukkan bahwa protokol HTTP ON dan protokol MQTT maupun CoAP OFF. Rata-rata pengukuran latency HTTP adalah 2,65 mili detik, sedangkan pengukuran *throughput* MQT adalah 4,623 kbps. Rata-rata nilai *packet loss* adalah sebesar 1%.

TABEL VII
PENGUJIAN DELAY DAN THROUGHPUT PADA HTTP SKENARIO I

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
5	8658	67.864	1.02	1.08	0%
10	16377	55.228	2.372	1.16	0%
15	24398	52.458	3.72	1.32	0%
20	32421	52.921	4.901	1.63	0%
25	40470	53.501	6.051	2.15	1%
30	48505	56.216	6.902	3.37	2%
35	56533	61.129	7.398	7.84	3%
AVERAGE			**4.623**	**2.65**	**1%**

Tabel VIII menunjukkan bahwa sistem hanya mengirimkan data melalui protokol CoAP, yang menunjukkan bahwa protokol CoAP ON dan protokol MQTT maupun HTTP OFF. Rata-rata pengukuran latency CoAP adalah 2,65 mili detik, sedangkan pengukuran *throughput* CoAP adalah 4,623 kbps. Rata-rata nilai *packet loss* adalah sebesar 1%.

TABEL VIII
PENGUJIAN DELAY DAN THROUGHPUT PADA COAP SKENARIO I

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
5	9610	72.86	1.055	0.87	0%
10	18078	52.771	2.74	0.98	0%
15	27811	55.93	3.977	1.17	0%
20	35832	52.331	5.477	1.46	0%
25	44688	52.235	6.844	2.01	2%
30	53576	52.672	8.137	2.92	2%
35	62443	54.066	9.239	7.58	3%
AVERAGE			**5.353**	**2.43**	**1%**

B. Skenario II (tiga protokol bersamaan)

Pengujian kinerja kedua dilakukan dengan skenario mengirimkan data sensor melalui ketiga protokol secara bersamaan.

Tabel IX menunjukkan bahwa sistem mengirimkan baik data dari protokol MQTT,

CoAP dan HTTP ON jadi, ketiga protokol terhubung ke Tidakde gateway. Rata-rata pengukuran *delay* multi-protokol MQTT adalah 1,62 mili detik dan pengukuran *throughput* multi-protokol MQTT adalah 5,554 kbps.

TABEL IX
PENGUJIAN DELAY DAN THROUGHPUT PADA MQTT SKENARIO II

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
10	17492	50.067	2.794	1.07	0%
15	26121	50.383	4.147	1.15	0%
20	34726	51.148	5.431	1.23	0%
25	43375	53.156	6.527	1.47	0%
30	51957	59.697	6.962	1.93	0%
35	60874	65.261	7.462	2.86	1%
AVERAGE			**5.554**	**1.62**	**0%**

Tabel X menunjukkan bahwa sistem mengirimkan dari ketiga protokol, jadi ketiganya terhubung ke Tidakde gateway. Rata-rata pengukuran *delay* multi-protokol HTTP adalah 1,95 milidetik dan pengukuran *throughput* multi-protokol HTTP adalah 4,921 kbps.

TABEL X
PENGUJIAN DELAY DAN THROUGHPUT PADA HTTP SKENARIO II

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
10	16376	55.424	2.363	1.08	0%
15	24401	62.541	3.121	1.16	0%
20	32419	70.313	3.688	1.32	0%
25	40470	53.501	6.051	2.17	2%
30	48505	56.216	6.902	2.56	2%
35	56533	61.129	7.398	3.38	3%
AVERAGE			**4.921**	**1.95**	**1.2%**

Tabel XI menunjukkan bahwa sistem mengirimkan dari ketiga protokol ON, jadi ketiganya terhubung ke Tidakde gateway. Rata-rata pengukuran *delay* multi-protokol CoAP adalah 1,71 mili detik dan pengukuran *throughput* multi-protokol CoAP adalah 5,732 kbps.

TABEL XI
PENGUJIAN DELAY DAN THROUGHPUT PADA COAP SKENARIO II

Jumlah Perangkat	Paket Terkirim (Bytes)	Time Span (s)	Throughput (kbit/s)	Delay (ms)	Packet Loss
10	18172	53.674	2.708	0.87	0%
15	26946	55.987	3.85	0.98	0%
20	35828	79.371	3.611	1.17	0%
25	44688	52.235	6.844	2.22	2%
30	53576	52.672	8.137	2.08	2%
35	62443	54.066	9.239	2.95	3%
AVERAGE			**5.732**	**1.71**	**1.2%**

IV. KESIMPULAN

Penelitian ini mengusulkan arsitektur gateway multi-protokol komunikasi untuk crowdsensing yang menggunakan perangkat dengan beragam protokol komunikasi. Kami menggabungkan ketiga jenis protokol komunikasi yaitu MQTT, HTTP dan CoAP yang dijalankan pada gateway. Gateway ini kemudian mengubah ketiga protokol tersebut ke dalam protokol yang sama dengan back-end server di cloud. Penelitian ini juga mengembangkan beberapa perangkat *crowdsensing* kesehatan multi-protokol dan platform kesehatan IoT berbasis cloud. Dari penelitian ini, semua protokol dapat dijalankan pada satu gateway sehingga perangkat dengan protokol yang berbeda dapat mengirimkan data. Selain itu juga dilakukan pengukuran kinerja ketiga protokol berdasarkan *delay*, *throughput*, dan *packet loss*. Secara umum, protokol MQTT memiliki performa terbaik dalam mengukur *delay*. Di sisi lain, protokol CoAP memiliki nilai *throughput* terbaik. Terdapat *packet loss* kurang dari 4% selama eksperimen semua protokol. Temuan eksperimental untuk setiap protokol menunjukkan bahwa nilai penundaan meningkat dengan jumlah perangkat yang digunakan. Hal ini dapat mempengaruhi kualitas pelayanan khususnya dalam pemantauan kesehatan. Oleh karena itu, perlu dilakukan peningkatan sistem pada gateway tersebut. Akan sangat bagus juga jika gateway dapat bekerja pada protokol komunikasi jenis lain, sensor, dan cara mendiagnosis penyakit secara akurat di masa mendatang.

REFERENSI

[1] V. Pilloni, "How data will transform industrial processes: Crowdsensing, crowdsourcing and big data as pillars of industry 4.0," *Futur. Internet*, 2018, doi: 10.3390/fi10030024.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, 2011, doi: 10.1109/MCOM.2011.6069707.

[3] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "SACRM: Social Aware Crowdsourcing with Reputation Management in mobile sensing," *Comput. Commun.*, 2015, doi: 10.1016/j.comcom.2015.01.022.

[4] F. Laws, C. Scheible, and H. Schütze, "Active learning with amazon mechanical turk," in *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2011.

[5] C. Ken and L. Xiaoying, "A Zigbee based mesh network for ECG monitoring system," in *2010 4th International Conference on Bioinformatics and Biomedical Engineering, iCBBE 2010*, 2010, doi: 10.1109/ICBBE.2010.5514693.

[6] T. Ludwig, T. Siebigteroth, and V. Pipek, "Crowdmonitor: Monitoring physical and digital activities of citizens during emergencies," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, doi: 10.1007/978-3-319-15168-7_51.

[7] S. Jovanović *et al.*, "A mobile crowd sensing application for hypertensive patients," *Sensors (Switzerland)*, 2019, doi: 10.3390/s19020400.

[8] M. Marjanovic, A. Antonic, and I. P. Zarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2799707.

[9] R. Pryss, J. Schobel, and M. Reichert, "Requirements for a flexible and generic api enabling mobile crowdsensing mHealth applications," in *Proceedings - 2018 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems, RESACS 2018*, 2018, doi: 10.1109/RESACS.2018.00010.

[10] A. Zambrano, M. E. Ortiz, M. Zambrano Vizuete, and X. Calderón, "Crowdsensing and MQTT Protocol: A Real-Time Solution for the Prompt Localization of Kidnapped People," in *Advances in Intelligent Systems and Computing*, 2020, doi: 10.1007/978-3-

030-32033-1_22.

[11] S. Maqbool, M. Waseem Iqbal, M. Raza Naqvi, K. Sarmad Arif, M. Ahmed, and M. Arif, "IoT Based Remote Patient Monitoring System," in *2020 International Conference on Decision Aid Sciences and Application, DASA 2020*, 2020, doi: 10.1109/DASA51403.2020.9317213.

[12] E. T. S. Institute, "Telecommunications and {Internet} Protocol Harmonization Over Networks ({TIPHON}); {End}-to-End Quality of Service in {TIPHON} Systems; {Part} 5: {Quality} of Service ({QoS}) measurement methodologies," *ETSI TS 101 329-5 v1.1.2*, 2002.

[13] R. Bank, "ITU - Telecommunication Standardization Sector," *Construction*. 2001.

[14] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," 2017, doi: 10.1109/SysEng.2017.8088251.

[15] "Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/. [Accessed: 22-Jan-2022].

[16] "paho-mqtt · PyPI." [Online]. Available: https://pypi.org/project/paho-mqtt/. [Accessed: 13-Feb-2022].

[17] "Wireshark · Go Deep." [Online]. Available: https://www.wireshark.org/. [Accessed: 25-Jan-2022].