Available at https://jtsiskom.undip.ac.id (3 October 2019)
DOI:10.14710/jtsiskom.7.4.2019.127-133

Jurnal Teknologi dan Sistem Komputer, 7(4), 2019, 127-133

Kinerja gateway berbasis XMPP untuk komunikasi perangkat IoT

Performance of XMPP-based gateway for IoT devices communication

Mahar Faiqurahman[*)], Muhammad Malik Madani, Denar Regata Akbi

Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Malang
Jl Raya Tlogomas No 246 Malang, Indonesia 65144

Cara sitasi: M. Faiqurahman, M. M. Madani, and D. R. Akbi, "Uji kinerja gateway berbasis XMPP untuk komunikasi perangkat IoT," *Jurnal Teknologi dan Sistem Komputer*, vol. 7 no. 4, pp. 127-133, 2019. doi: 10.14710/jtsiskom.7.4.2019.127-133, [Online].

Abstract – This study examines the performance of a communication gateway for IoT devices by utilizing the XMPP protocol so that these devices can be connected and communicate using the Internet. The sensor nodes, which are IoT devices, were implemented using NodeMCU connected to the DHT11 sensor module and LED lights to simulate the incoming data. Sensor nodes can communicate using the XMPP protocol gateway and process the request-response data. Gateway data transmission performance with size variations from 10-100 MB gets an average delay time of 9.3 ms, an average jitter of 0.00178 ms, and an average throughput of 161.4 kbps. The CPU usage parameter has an average increase of 12%, and memory usage tends to be constant when data transmission occurs.

***Keywords** – IoT; XMPP protocol; communication gateway*

Abstrak – Penelitian ini mengkaji kinerja gateway komunikasi untuk perangkat-perangkat IoT dengan memanfaatkan protokol XMPP agar perangkat ini dapat saling terhubung dan berkomunikasi menggunakan jaringan Internet. Perangkat IoT berupa sensor node yang diimplementasikan menggunakan NodeMCU yang terhubung dengan modul sensor DHT11 dan lampu LED sebagai simulasi bahwa ada data yang masuk. Perangkat IoT telah dapat berkomunikasi menggunakan gateway protokol XMPP dan melakukan proses request-response. Hasil pengujian kinerja gateway saat transmisi data dengan variasi ukuran dari 10-100 MB mendapatkan waktu tunda rata-rata 9,3 ms, jitter rata-rata jitter 0,00178 ms, dan throughput rata-rata 161,4 kbps. Parameter penggunaan CPU memiliki kenaikan rata-rata 12% dan penggunaan memori cenderung konstan pada saat terjadi transmisi data.

***Kata kunci** – IoT; protokol XMPP; gateway komunikasi*

[*)]Penulis korespondensi (Mahar Faiqurahman)
Email: mahar@umm.ac.id

I. PENDAHULUAN

Berbagai perangkat lunak dan infrastruktur jaringan telah banyak dikembangkan agar dapat mengoptimalkan penggunaan Internet yang semakin mengalami peningkatan, salah satunya adalah teknologi *Internet of Things* (IoT). IoT merupakan inovasi yang ada di dalam jaringan global dan memungkinkan adanya interaksi dan komunikasi *Machine-to-Machine* (M2M) [1], [2]. IoT dapat memberikan solusi bagi manusia untuk mengelola dan mengoptimasi penggunaan benda di sekitar, seperti perangkat sensor, perangkat yang memanfaatkan *Radio Frequency Identification* (RFID), s*mart watch*, *smart rings*, *smart TV*, dan perangkat cerdas lainnya menggunakan perantara jaringan Internet secara *remote* (pada lokasi yang berbeda) [3], [4].

Perangkat-perangkat IoT tersebut umumnya tidak dapat terhubung satu dengan lainnya secara langsung walaupun tersambung dengan Internet [5]. Hal ini disebabkan karena adanya keterbatasan yang dimiliki oleh protokol lapisan jaringan (*network*) dimana semua perangkat tersebut belum tentu memiliki alamat publik di Internet. Untuk mengatasi permasalahan tersebut, diperlukan suatu perantara komunikasi antar perangkat, salah satunya dalam bentuk *gateway*.

Protokol komunikasi diperlukan oleh *gateway* agar dapat berkomunikasi dengan perangkat IoT. Protokol ini dirancang untuk mengintegrasikan beberapa perangkat yang berbeda yang terhubung dengan Internet. Penggunaan protokol dalam IoT harus mempertimbangkan banyak hal, di antaranya adalah karakteristik perangkat yang digunakan, fitur yang dimiliki oleh protokol, keterbatasan sumber daya pada perangkat komputasi, dan sumber daya listrik yang digunakan [6], [7].

Protokol yang dapat digunakan untuk menghubungkan perangkat melalui jaringan Internet di antaranya adalah DDS, CoAP, AMQP, MQTT, dan XMPP yang masing-masing memiliki kelemahan dan kelebihannya [6]. Protokol DDS mendukung konsep publisher-subscriber yang bersifat *brokerless* sehingga pengiriman datanya dapat dilakukan secara multicast. Protokol CoAP didasarkan pada protokol REST yang berjalan di atas HTTP sehingga perlu ada manajemen *session* untuk mengidentifikasi koneksi antar perangkat.

Submitted: 11 February 2019; Revised: 13 September 2019; Accepted: 18 September 2019; Published: 31 October 2019

Protokol MQTT berbasis pada *publish/subscribe* untuk pengiriman dengan waktu tunda yang lebih kecil daripada protokol CoAP. Seperti MQTT, protokol AMQP berbasis *publish/subscribe* yang memiliki fitur jaminan reliabilitas pengiriman data dan fitur interoperabilitas. Dari keempat jenis protokol tersebut, tidak ada yang mendukung model pengiriman data secara *real-time*, seperti *streaming* data, padahal dalam implementasi IoT ada kemungkinan data dikirimkan secara *streaming*, misalnya pada perangkat kamera (visual).

Protokol *Extensible Messaging and Presence Protocol* (XMPP) atau protokol Jabber dapat mengatasi kebutuhan IoT karena mendukung pesan kecil dan latensi yang rendah, mendukung komunikasi model *request-response* dan *publish-subscribe*, dan mendukung pengiriman *data streaming* [8]. XMPP berbasis XML dan memiliki tingkat skalabilitas tinggi yang meyediakan arsitektur terdesentralisasi serta mendukung banyak ekstensi yang telah didefinisikan [9]. Protokol ini digunakan untuk pertukaran data dan informasi status (*presence*) secara *real-time* sehingga dapat digunakan untuk mengidentifikasi *state* (kondisi) suatu node [10].

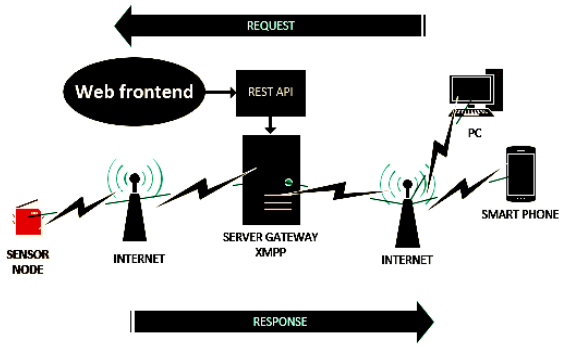
Protokol XMPP telah diimplementasikan sebagai media komunikasi pada *smart home object* [3], [11]. Sistem ini digunakan untuk pengelolaan, dan pengendalian perangkat rumah tangga dari jarak jauh. Pada penelitian yang lain juga telah diimplementasikan protokol XMPP untuk layanan komunikasi pada perangkat IoT [12]-[14]. Dari semua penelitian yang telah dilakukan tersebut, pemasangan *gateway* dilakukan hanya sebatas pada jaringan lokal saja, sehingga evaluasi kinerja komunikasi terbatas pada jaringan lokal. Kekurangan yang lain adalah skalabilitas perangkat IoT yang terhubung juga tidak tinggi.

Penelitian ini bertujuan membangun dan menguji kinerja *gateway* berbasis protokol XMPP untuk layanan komunikasi perangkat IOT. *Gateway* tersebut dibangun di atas *Virtual Private Server* (VPS) untuk meningkatkan skalabilitas koneksi dari perangkat IoT. Perangkat IoT harus terhubung dengan jaringan Internet agar dapat terkoneksi dengan *gateway* dan melakukan komunikasi dengan perangkat IoT yang lain.

II. METODE PENELITIAN

Penelitian ini menggunakan VPS yang mempunyai IP publik sebagai *gateway* komunikasi bagi perangkat IoT yang telah terintegrasi dengan protokol XMPP. Perangkat IoT harus terhubung ke jaringan Internet dan melakukan proses autentikasi agar dapat terhubung dengan *gateway*. Melalui *gateway*, pengguna dapat mengelola koneksi antar perangkat IoT sehingga dapat ditentukan perangkat IoT mana yang dapat saling berkomunikasi.

Dalam penelitian ini, perangkat IoT yang terhubung dengan *gateway* dapat melakukan pengiriman pesan *request* dan *response* ke perangkat lain. Jenis pesan yang dikirim tergantung dari perangkat IoT dan proses



Gambar 1. Arsitektur sistem keseluruhan

komputasi yang berjalan di dalamnya. Data disediakan oleh perangkat node sensor. Perangkat IoT yang membutuhkan data adalah berupa *smartphone* atau perangkat komputer. *Smartphone* atau perangkat komputer melakukan proses *request* data hasil penginderaan, sedangkan *sensor node* melakukan *response* berupa pengiriman data hasil penginderaan oleh sensor.

Node sensor diimplementasikan menggunakan perangkat NodeMCU yang terhubung dengan modul sensor DHT11. Lampu LED dalam node sensor digunakan sebagai simulasi bahwa ada data yang masuk. *Response* yang dikirimkan diharapkan sesuai dengan *request* yang diminta, baik berupa data hasil penginderaan menggunakan sensor DHT11, perintah untuk mematikan atau menyalakan lampu LED, maupun perintah lainnya yang telah disediakan pada program yang sudah ditanamkan di NodeMCU. Transmisi data antar perangkat IoT tersebut dilakukan melalui *gateway* yang memanfaatkan protokol XMPP sebagai media komunikasinya.

Arsitektur sistem *gateway* komunikasi yang diimplementasikan dinyatakan pada Gambar 1. Terdapat dua komponen utama sistem, yaitu *gateway* dan perangkat IoT. *Gateway* merupakan inti dari sistem yang dibangun sebagai penghubung antar perangkat IoT melalui protokol XMPP agar dapat saling berkomunikasi. Perangkat IoT merupakan perangkat yang dirancang untuk dapat saling berkomunikasi melalui *gateway*, yaitu berupa *smartphone*, komputer, dan node sensor yang diimplementasikan menggunakan NodeMCU dan sensor DHT11.

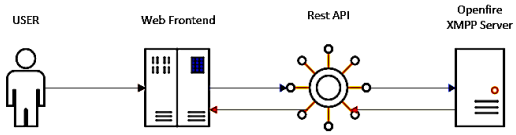
Data hasil penginderaan dikirimkan langsung secara *real-time* ke *smartphone* setelah melakukan *request*. Di dalam *gateway*, diimplementasikan server XMPP yang digunakan untuk mengatur koneksi dan pengelolaan perangkat IoT yang tergabung dalam sistem. XMPP server diimplementasikan menggunakan Openfire. Openfire menyediakan suatu API RESTful yang dapat digunakan untuk menghubungkan server XMPP dengan aplikasi yang lain. Dalam penelitian ini, API Restful digunakan untuk menghubungkan aplikasi *web* sebagai antarmuka pengguna dengan server XMPP sehingga pengguna dapat mengelola data yang ada di dalam server XMPP.

Node sensor (NodeMCU) diintegrasikan dengan beberapa modul *sensor* dan modul komunikasi agar dapat melakukan proses *sensing* dan pengiriman data. Di dalam memori internal node sensor juga dimasukkan program atau instruksi untuk melakukan proses penginderaan dan mengirimkan data hasil penginderaan. Modul sensor yang digunakan sebagai uji coba adalah sensor DHT11. Selain itu, node sensor juga dihubungkan dengan lampu LED untuk mensimulasikan adanya data yang masuk. Untuk menghubungkan node sensor dengan *gateway* melalui protokol XMPP, digunakan pustaka *XMPP client*. Pustaka tersebut memungkinkan node sensor menerima *request* dan memberikan *response* data hasil penginderaan dan data kontrol ke perangkat lain. Selain itu, node sensor juga dapat mengirimkan data hasil penginderaan dengan menggunakan mekanisme *push message* (tanpa didahului proses *request*) ke perangkat lain.

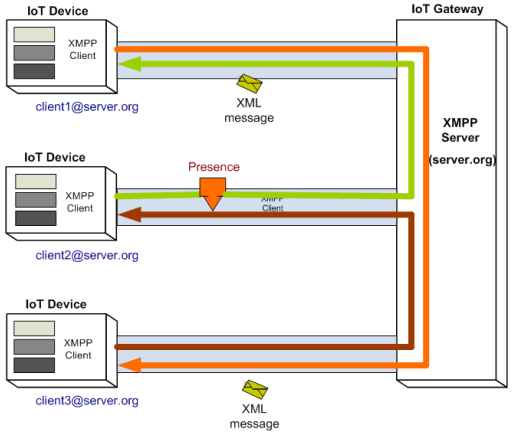
Smartphone disimulasikan sebagai perangkat yang melakukan *request* data hasil sensing ke node sensor. Perangkat ini juga digunakan untuk melakukan kontrol pada node sensor yang disimulasikan melalui LED yang terpasang di NodeMCU. Di dalam *smartphone* juga digunakan pustaka *XMPP client* yang berbasis pada sistem operasi Android. Pustaka *XMPP client* ini yang memungkinkan *smartphone* dapat terhubung dengan *gateway* dan melakukan pengiriman data *request* serta penerimaan data *response* dan *push message* dari node sensor.

Pengelolaan perangkat IoT yang terhubung dengan *gateway* menggunakan antarmuka aplikasi web. Aplikasi web ini merupakan aplikasi yang terpisah dengan server XMPP dimana komunikasi keduanya memanfaatkan *restful* API melalui protokol HTTP. Gambar 2 menunjukkan skema bagaimana pengguna mengaskses server XMPP melalui antarmuka web. Aplikasi web ini dibangun menggunakan bahasa pemrograman web. yaitu PHP, HTML, dan Javascript serta menggunakan database MySQL untuk media penyimpanannya. Aplikasi web ini mempunyai beberapa fitur, di antaranya melihat, membuat, menghapus, dan mengedit ID dari perangkat yang didaftarkan pada server XMPP, mengelompokkan ID perangkat ke beberapa kategori tertentu sesuai keinginan pengguna, dan manajemen pengguna (*user*) yang digunakan untuk mengatur akun pengguna pada aplikasi web.

Setiap perangkat yang terhubung dengan *gateway* memiliki ID yang unik (JID) sebagai identifikasi entitas. JID terdiri dari 3 bagian, yaitu *node identifier*, *domain*, dan *resource*. JID memiliki format penulisan seperti nama e-mail, yaitu *node@domain/resource* di mana ketiganya menjadi suatu pengenal (*identifier*) bagi setiap perangkat IoT yang terkoneksi dengan *gateway*. Setiap pengguna yang memiliki akses ke *gateway* dapat menambahkan lebih dari satu perangkat IoT yang dapat berkomunikasi dan masing-masing memiliki pengenal sendiri. Perangkat yang sudah ditambahkan oleh pengguna, dapat dikelompokkan ke dalam beberapa kelompok (*cluster*) yang berbeda sesuai dengan kategori



Gambar 2. Skema akses pengguna ke *gateway*

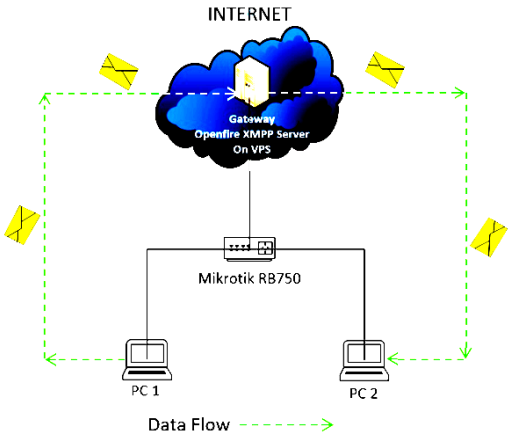


***Gambar 3**. XML stream* pada protokol komunikasi XMPP

aplikasi IoT. Proses pengelompokan ini memanfaatkan fitur *roster* yang terdapat di dalam XMPP.

Gambar 3 menunjukkan proses transmisi data yang berlangsung antar perangkat IoT melalui *gateway*. Format data yang dikirim antar perangkat IoT berupa format XML. Jenis XML yang ditransmisikan antar perangkat IoT melalui protokol XMPP, yaitu *message*, *presense*, dan *info query (IQ)*. *Message* berisi informasi data hasil penginderaan yang dikirimkan oleh suatu node sensor ke perangkat yang lain. *Presence* berisi informasi yang dikirimkan suatu perangkat IoT untuk mengetahui aktif tidaknya suatu perangkat IoT lain dalam sistem. Proses pengiriman dapat berjalan secara *broadcast* ke semua perangkat yang telah melakukan *subscribe* data yang diinginkan dari sensor node. *Info query* digunakan untuk mekanisme *request-response* antar perangkat IoT dalam jaringan XMPP, seperti metode *GET* dan *POST* pada protokol HTTP, dimana suatu perangkat IoT mengirimkan *request* ke perangkat yang lain untuk kemudian diberikan *response*.

Skenario pengujian sistem dilakukan dengan mengukur kinerja dari protokol XMPP ketika melakukan transmisi data antar perangkat IoT dimana kedua perangkat tersebut telah terautentikasi pada *gateway*. Gambar 4 merupakan rancangan skema alur komunikasi data untuk mengukur kinerja dari protokol XMPP saat mentransmisikan data antar perangkat. Pengukuran kinerja pada protokol XMPP menggunakan parameter nilai *delay*, *jitter*, dan *throughput* yang dianalisis dari hasil penangkapan paket data menggunakan aplikasi Wireshark saat transmisi berlangsung.



Gambar 4. Rancangan arsitektur untuk pengujian kinerja sistem

III. HASIL DAN PEMBAHASAN

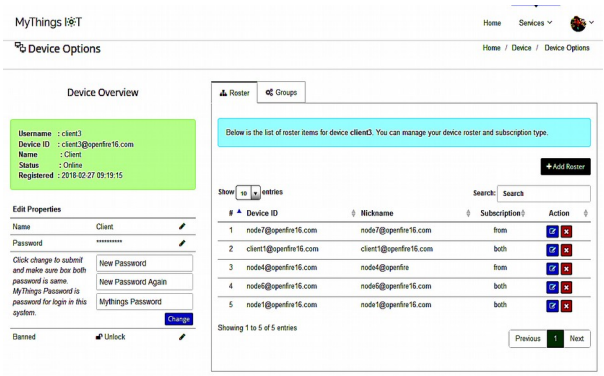
A. Aplikasi *front-end* berbasis w*eb*

Aplikasi *front end* berbasis *web* digunakan sebagai antarmuka pengguna untuk mengakses *gateway* dan digunakan sebagai pengelola perangkat IoT yang terhubung. Pengguna dalam aplikasi ini dapat melakukan pengaksesan dan pengelolaan perangkat IoT yang terhubung dengan *gateway*. Dengan menggunakan aplikasi ini, pengguna dapat menambahkan perangkat baru atau menghapus perangkat yang sudah terkoneksi sebelumnya. Setiap perangkat yang ditambahkan, diberikan ID yang merupakan JabberID dalam standar protokol XMPP. Pengguna juga dapat mengelompokkan perangkat-perangkat yang sudah ditambahkan sesuai dengan sistem IoT yang dibangun. Gambar 5 menunjukkan antarmuka pengguna dari *Web front end* yang telah dibuat.

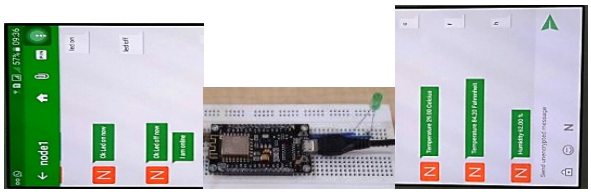
B. Pengujian *request-response* antar perangkat IoT

Pengujian proses *request-response* dilakukan dengan tujuan untuk memastikan apakah perangkat IoT dapat saling berkomunikasi dan berinteraksi satu dengan yang lainnya. Skenario pengujian dilakukan dengan cara melakukan proses pengiriman data *request* dari *smartphone* ke node sensor dimana keduanya telah terkoneksi dan terautentikasi ke *gateway*. Simulasi proses *request* yang dilakukan berupa pengiriman perintah untuk mematikan dan menghidupkan LED yang ada di node sensor. Skenario kedua dilakukan dengan proses pengiriman data hasil penginderaan dari node sensor ke *smartphone*.

Hasil dari pengujian proses *request-response* dapat dilihat pada Gambar 6. Gambar 6(a) merupakan proses *request* terhadap NodeMCU untuk menyalakan dan mematikan lampu LED. Gambar 6(b) merupakan kondisi lampu LED ketika kondisi menyala dan mati. Gambar 6(c) menunjukkan hasil penginderaan suhu dan kelembaban dengan menggunakan DHT11. Berdasarkan hasil pengujian tersebut, dapat dilihat bahwa proses pengiriman data dapat dilakukan oleh dua buah



Gambar 5. Salah satu antarmuka *front-end* web



(a) (b) (c)

Gambar 6. Pengujian proses *request* dan *response* terhadap perangkat IoT

perangkat (*smartphone* dan node sensor) melalui *gateway*.

C. Pengujian kinerja sistem

Pengujian ini bertujuan untuk mengukur kinerja dari protokol XMPP dalam menangani transmisi data antar perangkat dan kinerja dari *gateway* yang menjembatani transmisi data tersebut. Pada pengujian kinerja protokol XMPP dilakukan analisis nilai *delay*, *jitter*, dan *throughput* dengan menggunakan aplikasi Wireshark, sedangkan pada pengujian kinerja *gateway* dilakukan analisis *CPU usage* dan *memory usage* pada *gateway* saat transmisi antar perangkat IOT berlangsung.

Pengujian kinerja ini dilakukan dalam tiga skenario. Skenario pertama adalah pengiriman data dengan ukuran yang bervariasi dengan menggunakan nilai *transfer rate* yang konstan. Skenario kedua adalah pengiriman data dengan ukuran yang konstan dengan nilai *transfer rate* disetel bervariasi. Skenario ketiga adalah pengujian kinerja *gateway* pada saat kondisi *standby* dan pada saat transmisi data sedang berlangsung antar beberapa perangkat. Untuk setiap jenis pengujian, proses pengiriman data dimulai dari proses pembentukan koneksi antara perangkat IoT dengan *gateway*.

Untuk memudahkan penggunaan aplikasi Wireshark, perangkat IoT yang digunakan dalam pengujian ini berupa dua buah komputer yang keduanya terhubung dengan *gateway* melalui koneksi Internet. Untuk menghubungkan dua perangkat IoT tersebut dengan Internet dan *gateway*, digunakan *router* Mikrotik dan modem Internet pada jaringan lokal.

Pengujian protokol dengan variasi ukuran data.

Pada pengujian ini, ukuran data yang divariasikan antara 10 MB hingga 100 MB. *Transfer rate* dari Mikrotik RB750 ke modem dalam keadaan konstan yaitu 100 Mbps, dari modem ke ISP kecepatan *downlink* 77,08 Mbps dan *uplink* 21,85 Mbps, serta dari perangkat menuju Mikrotik RB750 memiliki *transfer rate* 100 Mbps. Pada *gateway* digunakan *bandwidth* 3,22 Mbps, *downlink* 3,26 Mbps, dan *uplink* 5,0 Mbps menuju perangkat. Nilai *delay*, *jitter*, dan *throughput* dengan variasi ukuran data transmisi tersebut dinyatakan dalam Gambar 7.

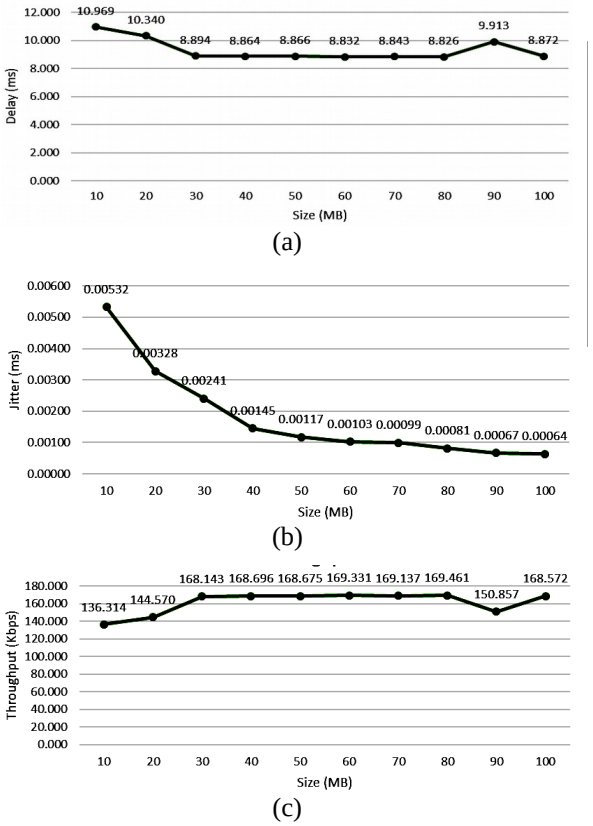
Hasil pengujian untuk nilai *delay* pada proses transmisi data menggunakan protokol XMPP dapat dilihat pada Gambar 7(a). Hasil pengujian di awal proses pengiriman data memiliki nilai *delay* yang lebih tinggi daripada data berikutnya, walaupun ukuran data yang dikirimkan lebih kecil dari data berikutnya. Hal ini disebabkan karena adanya mekanisme *handshaking* untuk pembentukan koneksi antara perangkat IoT dan *gateway* sebelum proses pengiriman data pertama dilakukan, sehingga *delay* yang dihasilkan lebih tinggi. Ketika koneksi telah terbentuk, maka delay pengiriman data akan cenderung konstan. Protokol XMPP bekerja dengan baik setelah koneksi dibuat karena pesan dikompresi menjadi berukuran kecil dan protokolnya sepenuhnya tersinkronisasi.

Hasil pengujian untuk nilai *jitter* pada proses transmisi data menggunakan protokol XMPP dapat dilihat pada Gambar 7(b). Sama halnya dengan hasil pengujian untuk nilai *delay*, adanya mekanisme *handshaking* pada awal pembentukan koneksi menjadikan nilai *jitter* pada pengujian di awal lebih tinggi dari pada data-data berikutnya. Ketika koneksi telah terbentuk, maka nilai *jitter* semakin menurun dan kinerja protokol XMPP semakin baik. Dari grafik tersebut, dapat dilihat bahwa penurunan nilai *jitter* saat transmisi data terus dilakukan. Hal ini menunjukkan semakin baiknya protokol XMPP dalam mentransmisikan data, meskipun ukuran data yang dikirim semakin besar.

Gambar 7(c) menunjukkan hasil pengujian nilai *throughput* pada proses transmisi data menggunakan protokol XMPP. Nilai *throughput* yang dihasilkan di awal ujicoba masih rendah, kemudian meningkat, dan cenderung tidak banyak perubahan untuk ukuran data di atas 30 MB. Faktor rendahnya *throughput* pada waktu awal ujicoba dipengaruhi oleh proses pembentukan koneksi di awal pengiriman data.

Pengujian protokol dengan variasi transfer rate

Pada pengujian ini, nilai *transfer rate* yang divariasikan adalah pada koneksi Mikrotik RB750 menuju ke modem yaitu mulai dari 0,25 Mbps hingga 100 Mbps. Ukuran dari data yang dikirim adalah konstan, yaitu sebesar 20 MB. Karena variasi *transfer rate* dilakukan pada jalur transmisi data perangkat IoT, maka hal ini akan berpengaruh terhadap *downlink* dan *uplink* dari PC/perangkat IOT menuju *gateway*. Hasil



Gambar 7. Kinerja *delay*, *jitter*, dan *throughput* transmisi untuk variasi ukuran data

pengujian *delay*, *jitter*, dan *throughput* terhadap variasi transfer rate ditunjukkan pada Gambar 8.

Hasil pengujian *delay* pada proses pengiriman data untuk variasi *transfer rate* menggunakan protokol XMPP dapat dilihat pada Gambar 8(a). Kenaikan nilai *delay* berbanding terbalik dengan kenaikan nilai *transfer rate*. Semakin besar *transfer rate* yang digunakan, maka semakin kecil nilai *delay* pengiriman data. Hal ini disebabkan karena semakin besar nilai *transfer rate* yang digunakan sehingga kapasitas pengiriman data juga semakin besar dan meminimalkan terjadinya *congestion* yang menyebabkan *delay*.

Gambar 8(c) menunjukkan hasil pengujian nilai *jitter* pada proses pengiriman data terhadap *variasi transfer rate*. Penurunan nilai *jitter* terjadi seiring dengan peningkatan nilai *transfer rate*, namun tidak terlalu signifikan. Seperti halnya dengan *delay*, peningkatan *transfer rate* menyebabkan potensi terjadinya *congestion* semakin kecil, sehingga nilai dari *jitter* semakin menurun.

Gambar 8(c) menunjukkan hasil pengujian *throughput* selama proses transmisi data dengan variasi *transfer rate*. Dari hasil pengujian tersebut, didapatkan bahwa nilai *throughput* meningkat seiring dengan peningkatan nilai *transfer rate*. Kenaikan tersebut cenderung stabil pada nilai *transfer rate* di atas 5 Mbps. Hal ini disebabkan karena semakin tinggi *transfer rate*, maka pontensi data untuk hilang pada saat pengiriman semakin rendah sehingga nilai *throughput* juga semakin tinggi.

Pengujian kinerja gateway

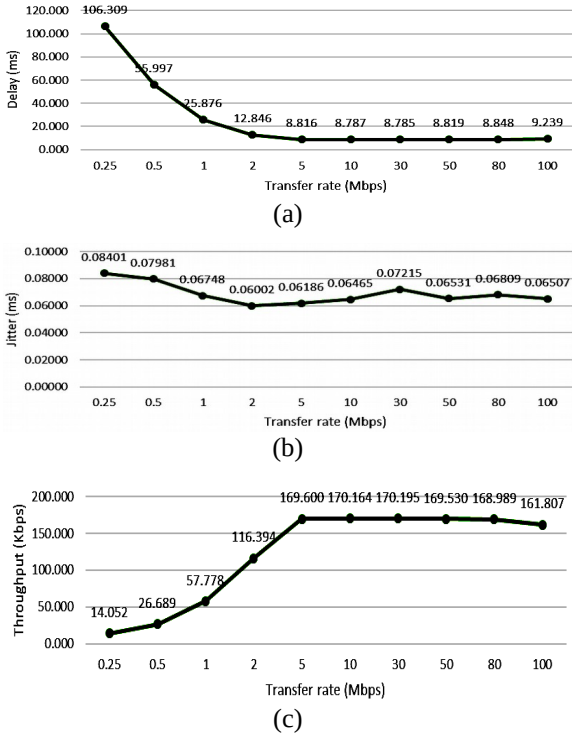
Pengujian ini dilakukan untuk mengetahui kinerja dari *gateway*. Kinerja yang diukur adalah persentase penggunaan CPU dan memori. Pada pengujian ini, ukuran data yang ditransmisikan adalah sebesar 20MB. Pengujian dilakukan dengan mengukur besar beban *gateway* saat beberapa perangkat terhubung sebagai *client* pada protokol XMPP. Perangkat yang terhubung divariasikan mulai mulai dari 2 hingga 20 perangkat. Pengujian dilakukan pada saat kondisi *standby* atau tidak ada aktivitas dan pada kondisi dimana perangkat melakukan transmisi data ke perangkat lain yang terhubung pada gateway. Selain itu, ukuran *transfer rate* juga bernilai konstan yaitu dari Mikrotik RB750 ke modem sebesar 100 Mbps, dari modem ke ISP untuk kecepatan *downlink* sebesar 77,08 Mbps, *uplink* sebesar 21,85 Mbps, dan dari perangkat menuju Mikrotik RB750 sebesar 100 Mbps. *Gateway* yang digunakan memiliki transfer rate sebesar 3,22 Mbps.

Gambar 9 menampilkan perbandingan persentase penggunaan CPU antara kondisi *standby* dengan kondisi ketika terjadi transfer data antar perangkat. Dari hasil pengujian tersebut, diperoleh bahwa persentase penggunaan CPU pada saat terjadi transfer data jauh lebih tinggi dari pada pada saat *standby*. Jika dilihat dari variasi jumlah perangkat yang terhubung, maka persentase penggunaan CPU pada kondisi *standby* cenderung konstan, sedangkan pada saat terjadi transfer data antar perangkat, penggunaan CPU cenderung fluktuatif. Rata-rata penggunaan CPU selama uji coba pada kondisi *standby* adalah sebesar 0,3%, sedangkan pada saat transmisi data adalah sebesar 12,3%.

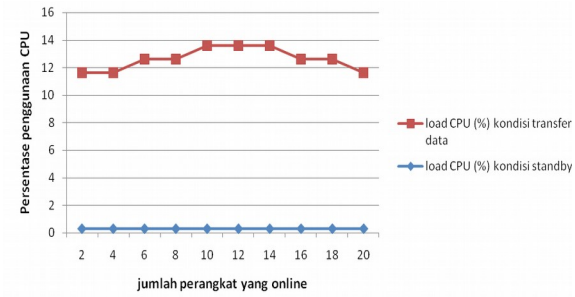
Perbandingan penggunaan memori pada *gateway* saat transmisi data dan saat *standby* ditunjukkan pada Gambar 10. Penggunaan memori pada saat transmisi data maupun pada saat *standby* cenderung sama. Jika dilihat dari variasi jumlah perangkat yang terhubung *gateway*, kenaikan jumlah perangkat tidak terlalu berpengaruh terhadap penggunaan memori. Rata-rata penggunaan memori pada saat pengujian adalah sebesar 8%.

Pada hasil pengujian kinerja *gateway* tersebut dapat dinyatakan bahwa terjadi kenaikan nilai persentase penggunaan CPU pada *gateway* saat transmisi berlangsung, namun kenaikan tersebut tidak terlalu tinggi dan cenderung konstan meskipun terdapat penambahan jumlah perangkat. Selama pengujian dilakukan, diperoleh bahwa persentase penggunaan memori tidak mengalami kenaikan, meskipun terjadi pada saat proses transmisi berlangsung.

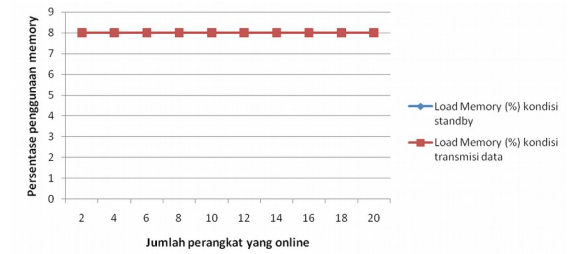
Secara umum dari hasil pengujian yang telah dilakukan, kinerja komunikasi pada protokol XMPP dalam penelitian ini menunjukkan nilai yang lebih rendah daripada penelitian sebelumnya [12]-[14], baik dari sisi nilai *delay*, *jitter*, dan *throughput*. Hal ini disebabkan karena pada penelitian sebelumnya, *gateway* dipasang pada jaringan lokal, sedangkan pada penelitian yang dilakukan ini *gateway* dipasang pada jaringan Internet yang memiliki *data rate* jauh lebih rendah dari pada jaringan lokal. Hal ini berpengaruh terhadap QoS yang dihasilkan oleh kedua sistem tersebut. Namun demikian, jika dibandingkan dengan [11]-[14], *gateway* yang dikembangkan dalam penelitian ini memiliki skalabilitas yang lebih tinggi dilihat dari jangkauan aksesnya secara publik.



Gambar 8. Kinerja *delay*, *jitter*, dan *throughput* transmisi untuk variasi *transfer rate*



Gambar 9. Perbandingan penggunaan CPU pada saat *standby* dan transfer data



Gambar 10. Perbandingan penggunaan memori pada saat *standby* dan transfer data

IV. KESIMPULAN

Gateway komunikasi pada perangkat IoT telah berhasil diimplementasikan dengan menggunakan protokol XMPP dengan akses publik lewat Internet. Protokol ini mempunyai kinerja yang cukup baik dengan nilai *delay* dan *jitter* yang tidak terlalu tinggi terhadap perubahan ukuran data maupun *transfer rate* yang digunakan, nilai *throughput* yang memadai, persentase penggunaan CPU pada saat transmisi data di bawah 15% untuk jumlah perangkat sampai 20, dan penggunaan memori cenderung konstan sebesar 80% pada saat transmisi berlangsung.

DAFTAR PUSTAKA

[1] M. U. Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A review on internet of things (IoT)," *International Journal of Computing Applications*, vol. 113, no. 1, pp. 1-7, 2015. doi: 10.5120/19787-1571

[2] R. Klauck and M. Kirsche, "hatty things-making the internet of things readily usable for the masses with XMPP," in *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Pittsburgh, USA, Oct. 2012, pp. 60-69. doi: 10.4108/icst.collaboratecom.2012. 250464

[3] Y. Wenbo, W. Quanyu, and G. Zhenwei, "Smart home implementation based on Internet and wifi technology," in *2015 34th Chinese Control Conference (CCC)*, Hangzhou, China, Jul. 2015, pp. 9072–9077. doi: 10.1109/ChiCC.2015. 7261075

[4] A. Junaidi, "Internet of things, sejarah, teknologi dan penerapannya: review," *Jurnal Ilmiah Teknologi Informasi Terapan*, vol. 1, no. 3, pp. 62-66, 2015.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communication Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. doi: 10.1109/COMST.2015.2444095

[6] P. Masek et al., "Implementation of true IoT vision: survey on enabling protocols and hands-on experience," *International Journal of Distributed Sensor Networks*, vol. 12, no. 4, 2016. doi: 10.1155/2016/8160282

[7] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, and M. Ameling, "A service infrastructure for the internet of things based on XMPP," in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, San Diego, USA, Mar. 2013, pp. 385–388. doi: 10.1109/ PerComW.2013.6529522

[8] P. Kayal and H. Perros, "A comparison of iot application layer protocols through a smart parking implementation," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris, France, Mar. 2017, pp. 331–336. doi: 10.1109/ICIN.2017.7899436

[9] M. B. Yassein, M. Q. Shatnawai, and D. Al-zoubi, "Application layer protocols for the internet of things: a survey," in *2016 International Conference on Engineering & MIS (ICEMIS)*, Agadir, Marocco, Sep. 2016, pp. 1-4. doi: 10.1109/ ICEMIS.2016.7745303

[10] E. Zuliarso and H. Februariyanti, "Pemanfaatan instant messaging untuk aplikasi layanan akademik," *Jurnal Dinamik*, vol. 18, no. 2, pp. 112-121, 2013.

[11] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: a smart home in a box," *Computer (Long Beach Calif)*, vol. 46, no. 7, pp. 62–69, 2013. doi: 10.1109/MC.2012.328

[12] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," in *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, Cairo, Egypt, Apr. 2016, pp. 1-7. doi: 10.1109/MoWNet.2016.7496622

[13] M. Pohl, J. Kubela, S. Bosse, and K. Turowski, "Performance evaluation of application layer protocols for the internet-of-things," in *2018 Sixth International Conference on Enterprise Systems (ES)*, Limassol, Cyprus, Oct. 2018, pp. 180-187. doi: 10.1109/ES.2018.00035

[14] H. Wang, D. Xiong, P. Wang, and Y. Liu, "A lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices," *IEEE Access*, vol. 5, pp. 16393–16405, 2017. doi: 10.1109/ ACCESS.2017.2742020