Kinerja Gateway Berbasis XMPP untuk Layanan Komunikasi pada Perangkat IoT

Performance of XMPP-Based Gateway for IoT Device Communication Services

Mahar Faiqurahman[*,1)], Muhammad Malik Madani[2)], Denar Regata Akbi[3)]

[1,2,3)] *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, " Kinerja Gateway Berbasis XMPP untuk Layanan Komunikasi pada Perangkat IoT" Jurnal Teknologi dan Sistem Komputer, vol. 7 no. 4, 2019. doi: 10.14710/jtsiskom.x.x.xxxx.xx-xx, [Online].

Abstract –In this study we build a communication Gateway to support inter-device connection and communication which is needed by IoT system. The purpose of this system, is to make the IoT device can connect each other using internet connection. From the experiment result we can see that the connecting IoT devices can communicate by sending request-response message. The experiment result also shows that during data transmission process, the CPU usage has increased for at least 12%, and memory usage has inclined stably.

Keywords *- — IoT, XMPP Protocol, Communication Gateway*

Abstrak - Dalam penelitian ini dibangun sebuah Gateway komunikasi untuk perangkat-perangkat IoT dengan memanfaatkan protokol XMPP. Tujauannya adalah agar perangkat-perangkat IoT yang ada dapat saling terhubung, dan berkomunikasi menggunakan jaringan internet. Berdasarkan hasil pengujian, didapatkan bahwa perangkat IoT yang ada, dapat berkomunikasi menggunakan gateway protokol XMPP dan melakukan proses request-response. Hasil pengujian performa saat transmisi dengan variasi ukuran data didapatkan rata-rata delay=9.3 ms, rata-rata jitter=0.00178 ms, dan rata-rata througput=161.4 kbps. Hasil pengujian juga menunjukan parameter penggunaan CPU memiliki kenaikan rata-rata 12%, serta penggunaan memori cenderung konstan pada saat terjadi transmisi data.

Kata kunci *- IoT, Protokol XMPP, Gateway Komunikasi*

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) yang 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), *Smart Watch*, *Smart Rings*, *Smart TV* dan perangkat cerdas lainnya menggunakan perantara jaringan internet secara *remote* (pada lokasi yang berbeda) [3][4].

Secara umum perangkat-perangkat IoT tersebut tidak dapat terhubung satu dengan lainnya secara langsung, walaupun perangkat-perangkat tersebut terhubung dengan internet [5]. Hal ini disebabkan karena adanya keterbatasan yang dimiliki oleh protokol *layer network*, dimana semua perangkat tersebut belum tentu memiliki alamat publik di internet. Untuk mengatasi permasalahana tersebut, diperlukan suatu perantara komunikasi antar perangkat, yang salah satunya dalam bentuk *Gateway*.

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

Beberapa protokol yang dapat digunakan untuk menghubungkan perangkat melalui jaringan internet diantaranya adalah DDS, CoAP, AMQP, MQTT, XMPP, dimana protokol-protokol tersebut memiliki kelemahan dan kelebihannya masing-masing [6]. Protokol DDS mendukung konsep publisher subscriber yang bersifat brokerless sehingga pengiriman datanya secara multicast. Protokol CoAP didasarkan pada porotokol REST yang berjalan diatas HTTP, sehingga perlu ada manajemen session untuk mengidentifikasi koneksi antar perangkat. Protokol MQTT merupakan protokol messaging yang yang berbasis pada publish/ subscribe untuk pengiriman dengan delay pengiriman yang lebih kecil daripada protokol CoAP. Protokol AMQP termasuk protokol publish/subscribe yang memiliki fitur

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

jaminan reliabilitas pengiriman data dan fitur interoperabilitas. Dari keempat jenis protokol tersebut, tidak ada yang mendukung model pengiriman data secara real-time seperti data streaming, sedangkan dalam implementasi IoT, ada kemungkinan data dikiriman secara streaming, misalkan pada perangkat kamera.

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

Pada penelitian sebelumnya pernah diimplementasikan protokol XMPP 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][13][14]. Dari semua penelitian yang telah dilakukan tersebut, *deployment Gateway* dilakukan hanya sebatas pada jaringan lokal saja, sehingga evaluasi performa komunikasi juga terbatas pada jaringan lokal. Kekurangan yang lain adalah skalabilitas perangkat IoT yang terhubung juga tidak tinggi.

Penelitian ini bertujuan untuk membangun dan mengukur kinerja *Gateway* berbasis protokol XMPP untuk layanan komunikasi perangkat IOT. Gateway tersebut dibangun diatas suatu *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.

III. METODE PENELITIAN

Dalam penelitian ini digunakan *Virtual Private Server* (VPS) yang mempunyai IP *public* 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.

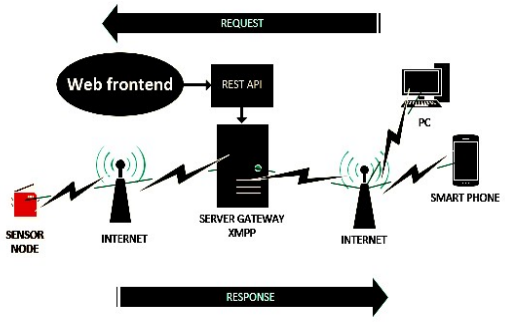
Dalam penelitian ini diasumsikan bahwa perangkat IoT yang terhubung dengan *Gateway* dapat melakukan pengiriman pesan *request* dan *response* ke perangkat lain. Jenis pesan yang dikirim tergantung dari perangkat IoT dan proses komputasi yang berjalan di dalamnya. Terdapat perangkat IoT yang menyediakan data, dalam hal ini diasumsikan berupa perangkat *sensor node*, dan perangkat IoT yang membutuhkan data yang berupa *smartphone* atau perangkat komputer. *Smartphone* atau perangkat komputer akan melakukan proses *request* data hasil *sensing*, sedangkan *sensor node* akan melakukan *response* berupa pengiriman data hasil *sensing* oleh sensor.

Sensor node diimplementasikan menggunakan perangkat NodeMCU yang terhubung dengan modul *sensor* DHT11 dan lampu LED sebagai simulasi bahwa ada data yang masuk. *Response* yang dikirimkan akan sesuai dengan *request* yang diminta, baik berupa data hasil *sensing* 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 dapat dilihat pada Gambar 1. Terdapat dua komponen utama sistem, yaitu *Gateway* dan perangkat IoT. *Gateway* merupakan inti dari sistem yang dibangun, yaitu sebagai penghubung antar perangkat IoT melalui protokol XMPP agar dapat saling berkomunikasi. Perangkat IoT merupakan perangkat yang dirancang untuk dapat saling berkomunikasi melalui *Gateway*, dalam hal ini berupa *smartphone*, computer, serta *sensor node* yang diimplementasikan menggunakan NodeMCU dan sensor DHT11.

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

Sensor node (NodeMCU) harus diintegrasikan dengan beberapa modul *sensor* dan modul komunikasi agar dapat melakukan proses *sensing* dan pengiriman data. Di dalam internal memory *Sensor node* juga harus dimasukan program atau instruksi untuk melakukan proses sensing, dan mengirimkan data hasil sensing. Modul sensor yang digunakan sebagai uji coba adalah sensor DHT11. Selain itu, *Sensor node* juga dihungkan dengan lampu LED untuk mensimulasikan adanya data yang masuk. Untuk menghubungkan *Sensor node* dengan *Gateway* melalui protokol XMPP, digunakan library XMPP client. Dengan menggunakan library tersebut, memungkinkan *Sensor node* menerima *request* dan memberikan *response* data hasil sensing maupun



Gambar 1. Arsitektur Sistem

data kontrol ke perangkat lain. Selain itu, *Sensor node* juga dapat mengirimkan data hasil *sensing* dengan menggunakan mekanisme *push message* (tanpa didahului proses *request*) ke perangkat lain.

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

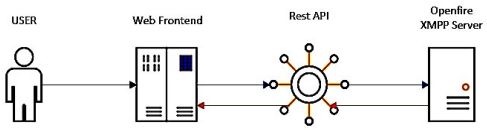
Pengelolaan perangkat IoT yang terhubung dengan *Gateway*, menggunakan antar muka aplikasi *web*. Aplikasi *web* ini merupakan aplikasi yang terpisah dengan XMPP *Server*, dimana komunikasi keduanya memanfaatkan *restful* API melalui protokol HTTP. Gambar 2 menunjukkan skema bagaimana pengguna mengaskses XMPP *Server* melalui antar muka *web*. Aplikasi *web* ini dibangun menggunakan Bahasa pemrograman *web* seperti PHP, HTML, Javascript serta menggunakan database MySQL untuk media penyimpanannya. Aplikasi *web* ini mempunyai beberapa fitur diantaranya:

- Melihat, membuat, menghapus, dan mengedit ID dari perangkat yang didaftarkan pada XMPP *Server* Openfire.
- Mengelompokkan ID perangkat ke beberapa kategori tertentu sesuai keinginan pengguna.
- Mengatur Roster untuk pengelompokan tiap perangkat serta menentukan tipe subscription pada perangkat, yaitu sebagai *publisher* atau *subscriber*.
- Manajemen pengguna (user) yang digunakan untuk mengatur akun pengguna pada aplikasi *web*.

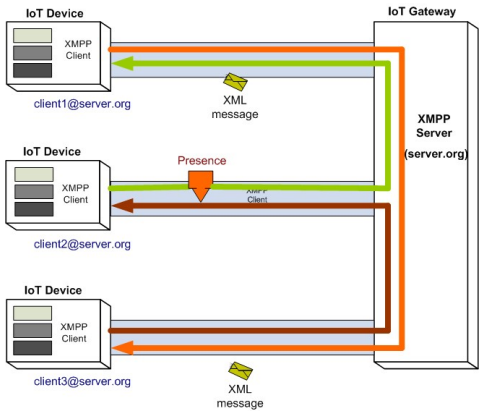
Setiap perangkat yang terhubung dengan Gateway memiliki ID yang unik yang disebut JID, yang merupakan fitur standar yang terdapat pada protokol XMPP sebagai identifikasi entitas. JID terdiri dari 3 bagian yaitu: *node identifier*, *domain*, dan *resource*. JID memiliki format penulisan seperti nama e-mail, yaitu *node@domain/resource*, dimana ketiganya akan 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, yang masing-masing memiliki pengenal sendiri. Perangkat yang sudah ditambahkan oleh pengguna, dapat pula dikelompokan ke dalam beberapa kelompok (*cluster*) yang berbeda sesuai dengan kategori aplikasi IoT. Proses pengelompokan ini memanfaatkan fitur *roster* yang terdapat di dalam XMPP.

Gambar 3 menunjukan bagaimana transmisi data berlangsung antar perangkat IoT melalui *Gateway*. Format data yang dikirim antar perangkat IoT berupa format XML. Terdapat 3 jenis XML yang umum ditransmisikan antar perangkat IoT melalui protokol XMPP yaitu:

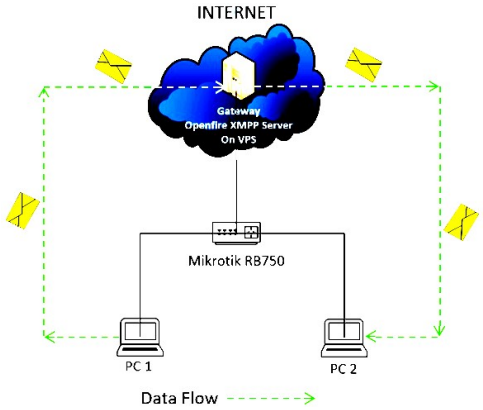
- *Message*, merupakan data utama dalam protokol XMPP. *Message* ini dapat berisi informasi data hasil *sensing* yang dikirimkan oleh suatu *sensor node* ke perangakat yang lain.
- *Presence*, merupakan informasi yang dikirimkan suatu perangkat IoT untuk mengetahui aktif tidaknya suatu perangkat IoT lain dalam sistem. Proses pengiriman dapat berjalan secara *broadcast* ke semua perangkat yang telah melakukan *subscribe* data yang diinginkan dari *sensor node*.



Gambar 2. Skema akses pengguna ke Gateway



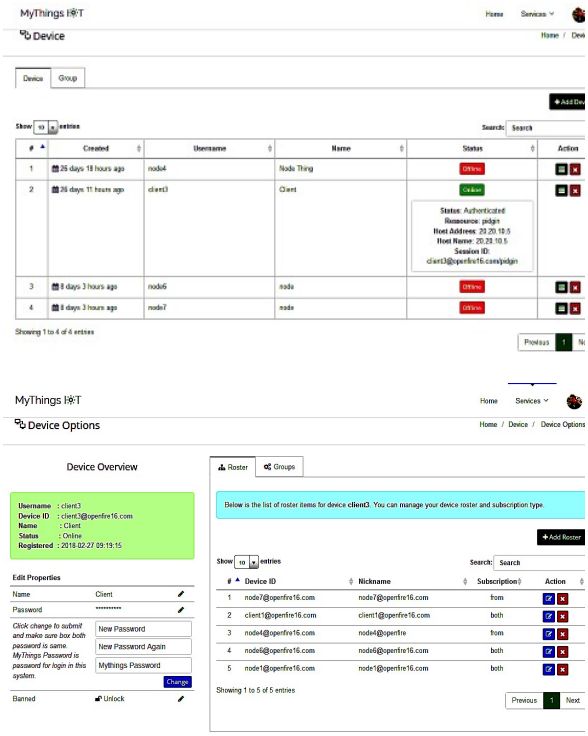
Gambar 3. XML Stream pada protokol komunikasi XMPP



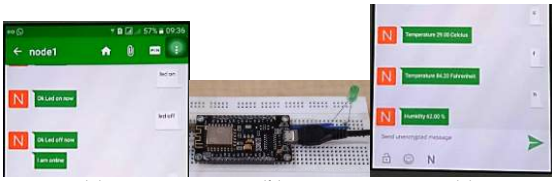
Gambar 4. Rancangan arsitektur untuk pengujian performa sistem

- *Info Query* (IQ) digunakan untuk mekanisme *request-response* antar perangkat IoT dalam jaringan XMPP seperti halnya metode *GET* dan *POST* pada protokol HTTP dimana suatu perangkat IoT akan mengirimkan *request* ke perangkat yang lain untuk kemudian diberikan *response*.

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



Gambar 5. Antar Muka *Web Front End*



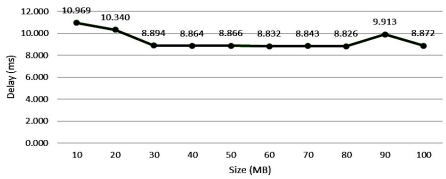
(a) (b) (c)

Gambar 6. Pengujian Proses *Request* dan *Response* terhadap Perangkat IoT

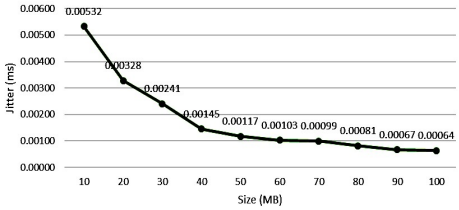
III. HASIL DAN PEMBAHASAN

A. Aplikasi *Web Front End*

Aplikasi *Web front end* merupakan aplikasi berbasis *web* yang digunakan sebagai antar muka pengguna untuk mengakses *Gateway,* yang 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 mengelompokan perangkat-perangkat yang sudah ditambahkan sesuai dengan sistem IoT yang akan dibangun. Gambar 5 merupakan antar muka pengguna dari *Web front end* yang telah dibuat.



Gambar 7. Grafik *delay* transmisi untuk variasi ukuran data



Gambar 8 Grafik *jitter* transmisi untuk variasi ukuran data

B. Pengujian *Request-Response* antar Perangkat IoT

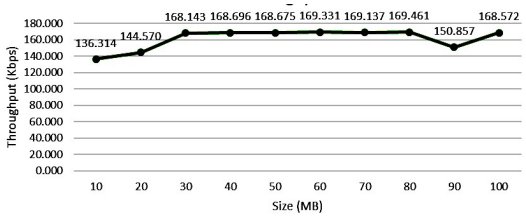
Pengujian proses *request-response* dilakukan dengan tujuan untuk memastikan apakah perangkat IOT dapat saling berkomunikasi dan berinteraksi antara satu dengan yang lain. Skenario pengujian dilakukan dengan cara melakukan proses pengiriman data *request* dari *smartphone* ke *sensor node,* dimana keduanya telah terkoneksi dan terautentikasi ke *Gateway*. Simulasi proses *request* yang dilakukan berupa pengiriman perintah untuk mematikan dan menghidupkan LED yang ada di *sensor node*. Kemudian skenario kedua dilakukan dengan proses pengiriman data hasil sensing dari *sensor node* 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 sensing suhu dan kelembaban dengan menggunakan DHT11. Berdasarkan hasil pengujian tersebut, dapat dilihat bahwa proses pengiriman data dapat dilakukan oleh dua buah perangkat (*smartphone* dan *sensor node*) melalui *Gateway*.

C. Pengujian Performa Sistem

Uji coba bertujuan untuk menguji kinerja dari dari protokol XMPP dalam menangani transmisi data antar perangkat, serta kinerja dari *Gateway* yang menjembatani transmisi data tersebut. Pada pengujian kinerja protokol XMPP dilakukan analisis nilai *delay*, *jitter*, *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.

Terdapat tiga skenario dalam pengujian performa. Pertama adalah pengiriman data dengan ukuran yang bervariasi, dengan asumsi menggunakan nilai *transfer rate* yang konstan. Kedua adalah pengiriman data dengan ukuran yang konstan, dengan asumsi nilai *transfer rate* dibuat bervariasi. Ketiga adalah pengujian kinerja *Gateway* pada saat kondisi *standby* dan pada saat transmisi data sedang berlangsung antar beberapa perangkat. Untuk setiap jenis pengujian diasumsikan bahwa proses pengiriman data dimulai dari proses pembentukan koneksi antara perangkat IoT dengan *Gateway*.



Gambar 9. Grafik troughput untuk variasi ukuran data

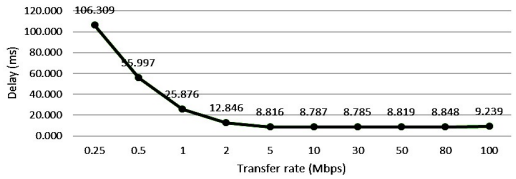
Untuk memudahkan penggunaan aplikasi Wireshark dalam pengujian ini diasumsikan bahwa perangkat IoT yang digunakan berupa dua buah komputer yang keduanya terhubung dengan *Gateway* melalui koneksi internet. Untuk menghubungkan dua perangkat IoT tersebut dengan internet dan *Gateway*, digunakan *router*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

Pengujian protokol dengan variasi ukuran data.

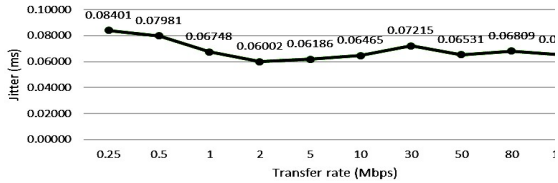
Pada pengujian ini ukuran data yang divariasikan antara 10 MB hingga 100 MB. Diasumsikan untuk *transfer rate* dari Mikrotik RB750 ke modem dalam keadaan konstan yaitu 100 Mbps, dari modem ke ISP kecepatan *downlink* 77.08 Mbps dan *uplink* 21.85 Mbps, serta dari perangkat menuju Mikrotik RB750 memiliki *transfer rate* 100 Mbps. Pada *Gateway* digunakan *bandwidth* 3.22 Mbps, *downlink* 3.26 Mbps, dan *uplink* 5.0 Mbps menuju perangkat.

Hasil pengujian untuk nilai *delay* pada proses transmisi data menggunakan protokol XMPP dapat dilihat pada Gambar 7. Dari grafik tersebut, didapatkan bahwa 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.

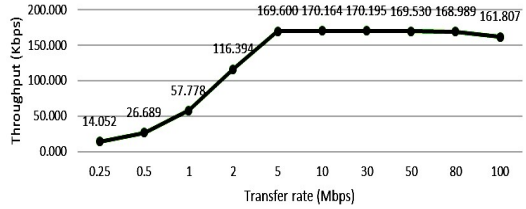
Hasil pengujian untuk nilai *jitter* pada proses transmisi data menggunakan protokol XMPP dapat dilihat pada Gambar 8. Sama halnya dengan hasil pengujian untuk nilai *delay,* adanya mekanisme *handshaking* pada awal pembentukan koneksi menjadikan nilai *jitter* pada pengujian di awal lebih tinggi dari pada data-data berikutnya. Ketika koneksi telah terbentuk, maka nilai *jitter* akan semakin menurun



Gambar 10. Grafik *delay* transmisi untuk variasi *transfer rate*



Gambar 11. Grafik *jitter* transmisi untuk *variasi transfer rate*



Gambar 12. Grafik *throughput* untuk variasi transfer rate

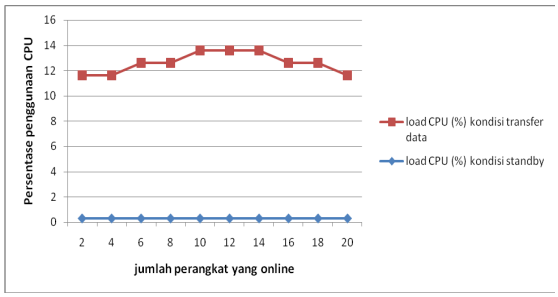
dan performa protokol XMPP akan semakin baik. Dari grafik tersebut dapat dilihat bahwa penurunan nilai *jitter* saat transmisi data terus dilakukan. Hal ini menunjukkan semakin baiknya protokol XMPP dalam mentransmisikan data, meskipun ukuran data yang dikirim semakin besar.

Gambar 9 menunjukan hasil pengujian nilai *throughput* pada proses transmisi data menggunakan protokol XMPP. Nilai *throughput* yang dihasilkan diawal ujicoba masih rendah, kemudian meningkat, dan cenderung tidak banyak perubahan untuk ukuran data diatas 30 MB. Faktor rendahnya throughput pada waktu awal ujicoba dipengaruhi oleh proses pembentukan koneksi di awal pengiriman data.

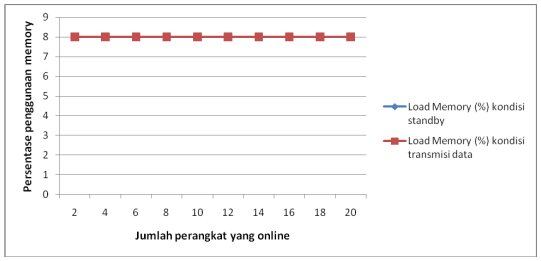
Pengujian protokol dengan variasi transfer rate

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 diasumsikan 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 pengujian *delay* pada proses pengiriman data untuk variasi *transfer rate* menggunakan protokol XMPP dapat dilihat pada Gambar 10. Dari grafik tersebut dapat dilihat bahwa kenaikan nilai *delay* berbanding terbalik dengan kenaikan nilai *transfer rate*. Semakin besar transfer rate yang digunakan, maka



Gambar 13. Grafik perbandingan penggunaan CPU pada saat *standby* dan transfer data



Gambar 14. Grafik perbandingan penggunaan memori pada saat *standby* dan transfer data

semakin kecil nilai delay pengiriman data. Hal ini disebabkan karena semakin besar nilai *transfer rate* yang digunakan, maka kapasitas pengiriman data juga semakin besar, sehingga meminimalkan terjadinya *congestion* yang menyebabkan *delay*.

Gambar 11 menunjukan hasil pengujian nilai *jitter* pada proses pengiriman data, dengan *variasi transfer rate*. Dari grafik tersebut dapat dilihat bahwa terjadi penurunan nilai *jitter* seiring dengan peningkatan nilai transfer rate, akan tetapi penurunan yang terjadi tidak terlalu signifikan. Seperti halnya dengan delay, hal ini disebabkan karena peningkatan *transfer rate* akan menyebabkan potensi terjadinya *congestion* semakin kecil, sehingga nilai dari jitter semakin menurun.

Gambar 12 menunjukan hasil pengujian *throughput* selama proses transmisi data dengan varias transfer rate. Dari hasil pengujian tersebut didapatkan bahwa nilai throuput meningkat seiring dengan peningkatan nilai *transfer rate*. Kenaikan tersebut cenderung stabil pada nilai *transfer rate* diatas 5 Mbps. Hal ini disebabkan karena semakin tinggi *transfer rate* maka pontensi data untuk hilang pada saat pengiriman akan semakin rendah, sehingga nilai throughput juga semakin tinggi.

Pengujian Kinerja Gateway

Pengujian ini dilakukan untuk mengetahui performa dari Gateway. Performa yang diukur adalah persentase penggunaan CPU dan memori. Pada pengujian ini, ukuran data yang transmisikan diasumsikan 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 akivitas, dan pada kondisi dimana perangkat melakukan transmisi data ke perangkat lain yang terhubung pada *Gateway*. Selain itu, ukuran *transfer rate* diasumsikan juga bernilai konstan yaitu dari Mikrotik RB750 ke modem sebesar 100 Mbps, dari modem ke ISP untuk kecepatan *downlink* sebesar 77.08 Mbps, dan *uplink* sebesar 21.85 Mbps, serta dari perangkat menuju Mikrotik RB750 sebesar 100 Mbps. *Gateway* yang digunakan memiliki *transfer rate* 3.22 Mbps.

Gambar 13 menampilkan perbandingan persentase penggunaan CPU antara kondisi *standby* dengan kondisi ketika terjadi transfer data antar perangkat. Dari hasil pengujian tersebut didapatkan bahwa persentase pernggunaan 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, dapat dilihat pada Gambar 14. Dari gambar tersebut dapat dilihat bahwa 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 diatas dapat dilihat bahwa terjadi kenaikan nilai persentase penggunaan CPU pada *Gateway* saat transmisi berlangsung, akan tetapi kenaikan tersebut tidak terlalu tinggi, dan cenderung konstan meskipun terdapat penambahan jumlah perangkat. Selama pengujian dilakukan, didapat bahwa persentase penggunaan memori tidak mengalami kenaikan, meskipun pada saat proses transmisi berlangsung.

Secara umum dari hasil pengujian yang telah dilakukan, performa komunikasi pada protokol XMPP dalam penelitian ini menunjukan nilai yang lebih rendah dari pada penelitian sebelumnya [12][13][14], baik dari sisi nilai *delay*, *jitter* dan *thoroughput*. Hal ini disebabkan karena pada penelitian sebelumnya, *Gateway* di-*deploy* pada jaringan lokal, sedangkan pada penelitian yang dilakukan, *Gateway* di-*deploy* pada jaringan internet yang memiliki *data rate* jauh lebih rendah dari pada jaringan lokal. Hal ini tentu saja berpengaruh terhadap QoS yang dihasilkan oleh kedua sistem tersebut. Namun demikian, jika dibandingkan dengan penelitian sebelumnya [11][12][13][14], *Gateway* yang dikembangkan dalam penelitian ini memiliki skalabilitas yang jauh lebih tinggi dari pada yang dikembangkan pada penelitian sebelumnya.

IV. KESIMPULAN

Gateway komunikasi pada perangkat IoT telah berhasil diimplementasikan dengan menggunakan protokol XMPP. Berbagai perangkat yang berbeda platform dapat saling berkomunikasi memanfaatkan *Gateway* yang berjalan diatas protokol XMPP. Dari hasil pengujian performa didapatkan bahwa protokol ini mempunyai performa yang cukup baik. Hal ini dibuktikan dari nilai *delay* dan *jitter* yang tidak terlalu tinggi untuk perubahan ukuran data maupun transfer rate yang digunakan. Begitu juga nilai *throughput* yang dihasilkan selama proses pengiriman data juga bagus. Persentase penggunaan CPU pada saat transmisi data mengalami fluktuasi pada saat pengiriman data, sedangkan penggunaan memori cenderung konstan pada saat transmisi berlangsung maupun ketika terjadi peningkatan jumlah perangkat IoT yang terhubung dengan *Gateway*.

DAFTAR PUSTAKA

[1] M. U. Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A review on Internet of Things (IoT)," *Int. J. Comput. Appl.*, vol. 113, no. 1, pp. 1–7, 2015.

[2] R. Klauck and M. Kirsche, "Chatty things-Making the Internet of Things readily usable for the masses with XMPP," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference On*, 2012, pp. 60–69.

[3] Y. Wenbo, W. Quanyu, and G. Zhenwei, "Smart home implementation based on Internet and WiFi technology," in *Control Conference (CCC), 2015 34th Chinese*, 2015, pp. 9072–9077.

[4] A. Junaidi, "Internet of Things, Sejarah, Teknologi dan Penerapannya," *J. Ilm. Teknol. Inf. Terap.*, vol. 1, no. 3, 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 Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] P. Masek *et al.*, "Implementation of true IoT vision: survey on enabling protocols and hands-on experience," *Int. J. Distrib. Sens. Networks*, vol. 12, no. 4, p. 8160282, 2016.

[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 *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, 2013, pp. 385–388.

[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)*, 2017, pp. 331–336.

[9] M. B. Yassein and M. Q. Shatnawi, "Application layer protocols for the Internet of Things: A survey," in *Engineering & MIS (ICEMIS), International Conference on*, 2016, pp. 1–4.

[10] E. Zuliarso and H. Februariyanti, "Pemanfaatan instant messaging untuk aplikasi layanan akademik," *Din. Teknol. Inf.*, vol. 18, no. 2, 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.

[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)*, 2016, pp. 1–7.

[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)*, 2018, pp. 180–187.

[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.