REPOSITOR, Vol. 2, No. 1, Januari 2020, Pp. 41-52
ISSN : 2714-7975
E-ISSN : 2716-1382

Rancang Bangun Gateway Komunikasi Pada Perangkat IoT Dengan Memanfaatkan Protokol XMPP

Muhammad Malik Madani[*1], Mahar Faiqurahman[2], Denar Regata Akbi[3]
[1,2,3]Teknik Informatika Universitas Muhammadiyah Malang
danimalik.m@gmail.com[*1], maharf@gmail.com[2], dnarregata@umm.ac.id[3]

Abstrak

IOT sebagai infrastruktur yang dirancang untuk memungkinkan benda-benda di sekitar kita dapat berkomunikasi satu sama lain, berfikir, bertindak, dan mengambil keputusan hanya dengan menggunakan perantara internet. Benda-benda tersebut tidak secara langsung dapat saling berkomunikasi dan dikendalikan, dibutuhkan suatu mekanisme yang dapat menangani jalur komunikasi serta dibangun khusus sebagai Gateway bagi lalu lintas komunikasi data. XMPP (Extensible Messaging and Presence Protocol) sebagai media komunikasi yang mendukung infrastruktur IOT dimana Protokol ini berbasis XML open-source dan mendukung banyak ekstensi yang telah di definisikan serta komunikasi real-time antar perangkat yang terdaftar pada jaringan protokol XMPP. Dalam penelitian ini dirancang sebuah gateway dengan memanfaatkan protokol XMPP agar perangkat IOT dapat saling berkomunikasi. Berdasarkan pengujian perangkat dapat berkomunikasi menggunakan gateway protokol XMPP dan melakukan proses request-response. Pada pengujian performansi saat transmisi diketahui bahwa pada kondisi transfer rate yang konstan dan ukuran data yang bervariasi memiliki rata-rata delay 9.322 ms, jitter 0.00178 ms, dan throughput 161.376 Kbps sedangkan pada kondisi transfer rate yang bervariasi dan ukuran data yang konstan memiliki rata-rata delay 25.432 ms, jitter 0.06885 ms, dan throughput 122.520 Kbps. Load server menunjukkan rata-rata penggunaan CPU 0.3% dan memory 8.0% saat perangkat dalam kondisi standby dan rata-rata penggunaan CPU 12.3% dan memory 8.0% saat transmisi data antar perangkat berlangsung. Kestabilan transmisi data dipengaruhi oleh kecepatan link antara perangkat menuju server sebagai gateway komunikasi pada protokol XMPP.

Kata Kunci: *IOT, Protokol, XMPP, Real-time, Gateway*

Abstract

IOT as an infrastructure designed to enable objects around us to communicate with each other, think, act, and use only the internet. They can not communicate with each other and are controlled, according to what can be used for cross-data communication. XMPP (Extensible Messaging and Presence Protocol) as a communications medium that supports IOT infrastructure where the protocol is XML-based open-source and supports many defined extensions and real-time communication between devices that record on the XMPP protocol network. In this study designed a gateway using XMPP protocol for IOT devices to communicate with each other. They can use the XMPP protocol gateway and perform the request-response process. At the moment the transmission performance is performed at a constant transfer speed and the data size varies on average of delay of 9,322 ms, jitter 0,00178 ms, and throughput of 161,376 Kbps while at different transfer rate and constant data size has average delay 25.432 ms, jitter 0.06885 ms, and throughput of 122 520 Kbps. The average server load uses CPU 0.3% and 8.0% of the memory when the device is in standby and average 12.3% CPU usage and 8.0% of the memory during data transmission between devices. The stability of data transmission by link speed between device to server as communication gateway on XMPP protocol.

Keywords: *IOT, Protocol, XMPP, Real-time, Gateway*

1. Pendahuluan

Berbagai software dan hardware telah banyak di kembangkan agar mempermudah manusia akan penggunaan internet yang semakin mengalami peningkatan. Untuk menghadirkan inovasi teknologi yang potensial, *Internet of Things* atau IoT hadir dan dikembangkan sebagai jaringan komputasi global yang memungkinkan adanya *human-machine communication* (H2M)

[1] [2]. IoT diharapkan merupakan solusi bagi manusia untuk mengelola dan mengoptimasi penggunaan benda di sekitar seperti perangkat sensor, *Radio Frequency Identification* (RFID), *Smart Watch, Smart Rings, Smart TV* dan *smart object* lainnya menggunakan perantara jaringan internet [3] [4].

Saat ini *Internet of Things* telah banyak diaplikasikan untuk meningkatkan kualitas hidup manusia. Aplikasi ini termasuk transportasi, agrikultur, *smart home*, kesehatan, industri dan beberapa kegiatan lain yang menjadi aktivitas manusia sehari-hari. Protokol di desain untuk mengintegrasikan hal-hal yang menjadi standar protokol internet mengingat perhitungan keterbatasan sumber daya memori, keterbatasan *bandwidth* dan ketersediaan energi juga menjadi pertimbangan dalam membangun sistem berbasis IoT [5] [6].

Sebagai sarana komunikasi bagi IoT tentunya dibutuhkan protokol yang mendukung karakteristik kebutuhan sensing dan komunikkasi data, beberapa protokol yang dapat digunakan untuk menerapkan gagasan *Internet of Things* dan membangun konektivitas antara *Smart Object* diantaranya DDS, CoAP, AMQP, MQTT, XMPP, HTTP, REST, SIP dimana protokol-protokol tersebut memiliki kelemahan dan kelebihannya masing-masing [7].

Extensible Messaging and Presence Protocol (XMPP) atau *Jabber Protocol* 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. Serte mendukung komunikasi model *request-response* dan *publish-subscribe.* XMPP memiliki tingkat skapabilitas tinggi yang meyediakan arsitektur terdesentralisasi dan mendukung banyak ekstensi yang telah di definisikan [8].

XMPP protokol juga banyak diterapkan pada aplikasi *Instant Messaging* sebagai fasilitas komunikasi *Chatting* bagi pengguna internet. Dengan menggunakan fasilitas ini user dapat berkomunikasi dengan pesan berupa text dan bertukar file secara *peer-to-peer.* Protokol ini berbasis XML untuk pertukaran *message* dan *presence* secara *real-time* dan *open source* sehingga pengembang dapat menggunakan sesuai dengan keinginannya [9].

Penelitian pernah dilakukan dengan menggunakan protokol XMPP sebagai media komunikasi pada *Smart Home Object* untuk mengelola perangkat rumah tangga untuk kemanan dan mengendalikan piranti dari jarak jauh. Dibangun suatu server lokal yang di remote dengan menggunakan protokol XMPP. Server tersebut dapat bertindak sebagai penyedia layanan dan menyediakan layanan untuk berbagai perangkat rumah dan kantor yang berbeda. [10]

Berdasarkan latar belakang tersebut tersebut, maka dalam penelitian ini dibuat *gateway* untuk perangkat IOT dengan memanfaatkan XMPP sebagai *protocol* komunikasi untuk membangun layanan *real-time communication* yang menggunakan sensor atau *smart object* sebagai media uji. Selain itu dilakukan pengujian untuk mengetahui kinerja dari *protocol* XMPP pada innfrastruktur IoT saat transmisi data berlangsung.

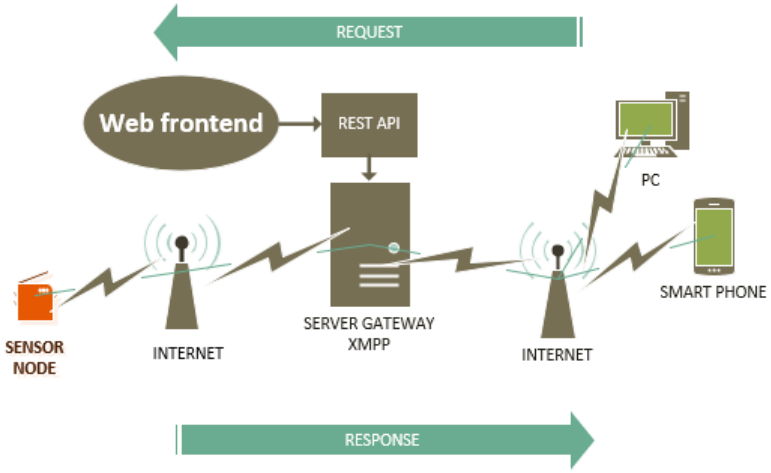
2. Metode Penelitian

Penelitian ini memerlukan *Virtual Private Server* sebagai gateway komunikasi bagi perangkat IoT yang telah terkonfigurasi dengan protokol XMPP. Untuk dapat terhubung dengan server maka perangkat IoT hanya membutuhkan koneksi ke jaringan internet dan melakukan proses autentikasi karena server dapat diakses dengan jaringan publik. Dengan begitu perangkat dapat saling terkoneksi dimana pun dan kapan pun.

Pada sistem yang dibangun ini, antar perangkat IoT akan melakukan proses *request-response.* Yang akan melakukan proses *request* adalah perangkat Smartphone ataupun Komputer yang telah terpasang aplikasi untuk XMPP client, sedangkan yang akan melakukan proses *response* yaitu *sensor node* yang menggunakan NodeMCU dan sensor DHT11 dan lampu LED. *Response* yang dikirimkan akan sesuai dengan *request* yang diminta, baik hasil data dari sensing menggunakan sensor DHT11, perintah untuk mematikan atau menyalakan lampu LED, dan perintah lainnya yang telah disediakan pada program di NodeMCU. Transmisi data antar perangkat IoT akan menggunakan protokol XMPP sebagai media komunikasinya.

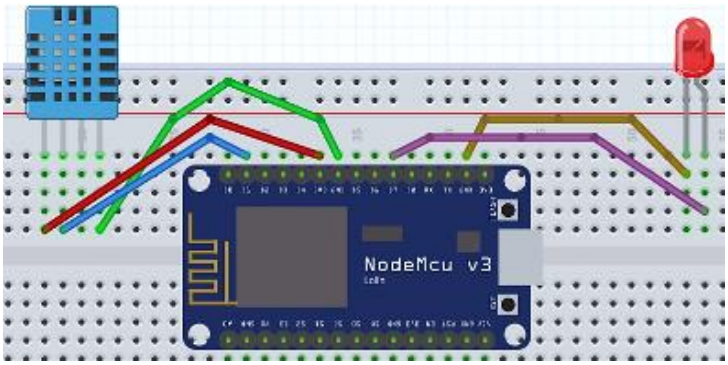
Gambar 1 mengilustrasikan arsitektur sistem yang diterapkan pada penilitian ini. Pada skenario yang diterapkan terdapat 2 komponen utama yaitu server gateway dan perangkat IoT. Server gateway merupakan inti dari jaringan protokol XMPP sebagai penghubung antar perangkat IoT dan agar dapat saling berkomunikasi. Sedangkan perangkat IoT merupakan perangkat untuk melakukan proses sensing yang dilakukan oleh *sensor node* menggunakan NodeMCU dan sensor DHT11 dan menerima hasil sensing oleh perangkat *smartphone*. Data

hasil sensing akan dikirimkan langsung secara *real-time* ke *smartphone* yang melakukan *request.* Pada server gateway telah dibangun aplikasi *web frontend* yang dibangun menggunakan Resfull API untuk mengatur ID device yang digunakan pada perangkat IoT untuk melakukan autentikasi pada server XMPP.



Gambar 1. Arsitektur Sistem

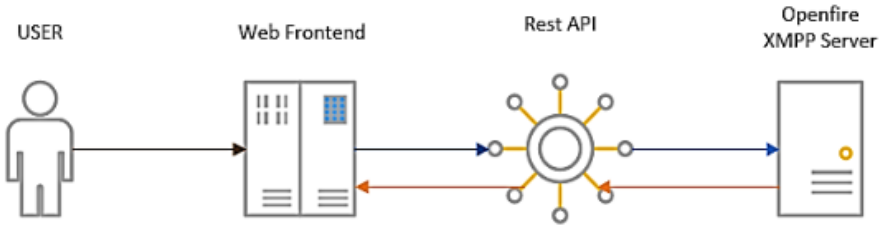
Agar dapat mengelola sensor dan menjalankan intruksi dan memberikan *response* sesuai *request* yang diberikan, digunakan NodeMCU sebagai *sensor node* yang digunakan untuk melakukan proses *sensing* dengan menggunakan sensor DHT 11 dan melakukan perintan untuk menyalakan dan mematikan lampu LED. Gambar 2 merupakan rangkaian pada NodeMCU serta penggunaan pin nya untuk input dan output data dari sensor DHT11 dan lampu LED. *Sensor node* melakukan proses *sensing* dan menjalankan perintah apabila ada *request* dari perangkat. Ketika tidak ada perinntah maka perangkat akan dalam kondisi idle menunggu perintah.



Gambar 2. Rangkaian pada NodeMCU

User tidak langsung mengakses server melainkan melalui aplikasi web yang telah dibuat khusus untuk menggunakan fitur Openfire XMPP server dengan menggunakanan RestAPI untuk POST dan GET data pada server sesuai kebutuhan selama penelitian yang dilakukan. Gambar 3 menunjukkan skema bagaimana user mengaskses server. Web frontend ini dibangun menggunakan Bahasa pemrograman web seperti PHP, HTML, Javascript serta menggunakan database MySQL untuk media penyimpanannya. Web frontend ini memberikan beberapa fitur diantaranya:

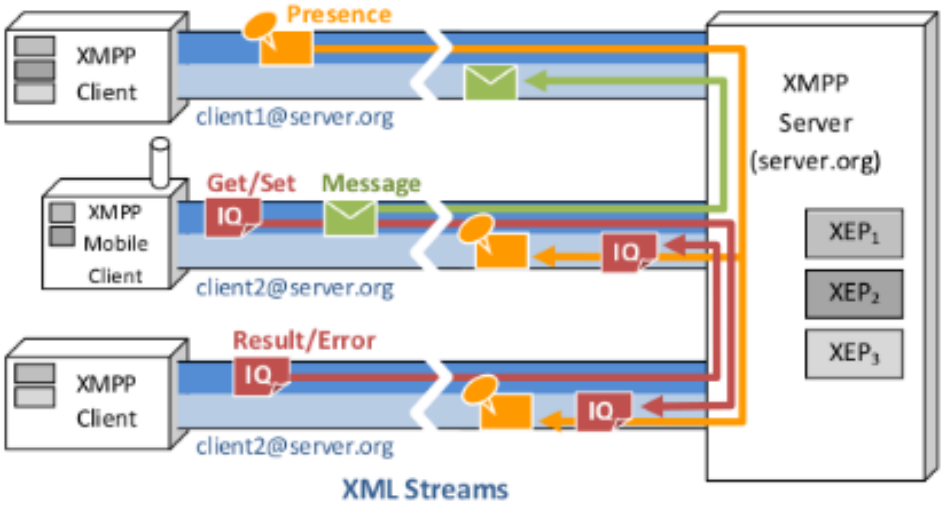
1. Melihat, membuat, menghapus, dan mengedit ID untuk perangkat yang di daftarkan pada server Openfire XMPP.
2. Mengelompokkan ID device ke beberapa kategori tertentu sesuai keinginan user.
3. Mengatur *Roster* dari tiap perangkat serta menentukan *Subscription Type* nya. Yaitu sebagai *publisher* atau *subscriber* antar perangkat.
4. User *management* untuk mengatur akun user pengguna pada sistem Web Frontend.



Gambar 3. Skema Akses User ke Server

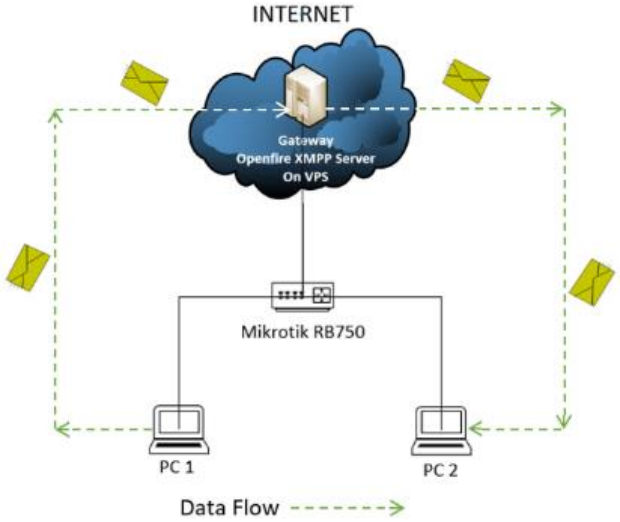
Pada protokol XMPP data yang ditransmisikan menggunakan format XML. Setiap entitas memiliki ID yang unik yang disebut *JabberID atau JID* yang terdiri dari 3 bagian diantaranya *local port, domain,* dan *resource.* Sebagai contoh "user@server.example.com/athome" dimana "user" merupakan *local port,* "server.example.com" merupakan *domain,* dan "athome" yang merupakan *resource session* dimana komunikasi sedang berlangsung. Pada Gambar 4 menggambarkan bagaimana transmisi data berlangsung antara client-server dan client-client. Terdapat 3 jenis XML yang umum ditransmisikan pada protokol XMPP antara lain:

1. **Message**, paket utama dalam jaringan XMPP, berisi informasi yang dikirimkan suatu entitas ke entitas lainnya yang bersifat *fire and forget,* artinya entitas pengirim tidak mendapatkan *result* dari paket yang dikirimkan ke entitas lain. Pengiriman bersifat *one-to-one* atau *one-to-many* entitas.
2. **Presence**, paket yang dikirimkan untuk mengetahui kehadiran dari tiap entitas yang terhubung dalam jaringan sehingga dapat diketahui status *online* atau *offline* setiap entitas. Paket yang dikirimkan ke tiap entitas bersifat *broadcast* ke semua entitas dalam jaringan yang telah *subscribe* ke entitas tersebut.
3. **Info Query (IQ)** digunakan untuk mekanisme *request-response* antar entitas dalam jaringan XMPP seperti halnya metode GET dan POST pada protokol HTTP dimana entitas yang mengirimkan *request* ke entitas lain akan mendapatkan *response* dari entitas tersebut.



Gambar 4. XML Stream Protokol Komunikasi XMPP

Dalam penelitian ini juga dilakukan pengukuran performansi dari protokol XMPP dengan melakukan transmisi data antar perangkat dimana kedua perangkat tersebut telah terinstall aplikasi untuk XMPP client dan telah melakukan authentikasi pada server gateway. Gambar 5 merupakan rancangan dan alur data untuk mengukur kinerja dari protokol XMPP saat mentransmisikan data antar perangkat. Untuk mengukur kinerja dari protokol XMPP dapat dilihat dari nilai *delay, jitter* dan *throughput* yang di analisa dari hasil capturing data menggunakan tools wireshark saat transmisi berlangsung sehingga data yang didapatkan menjadi lebih valid dan menganalisis load server saat menangani layanan protokol XMPP.

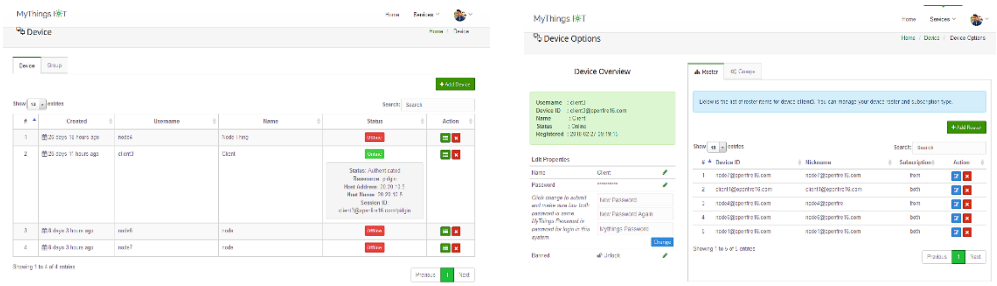


Gambar 5. Rancangan dan Alur Data Pengukuran Performansi Protokol XMPP

3. Hasil Penelitian dan Pembahasan

3.1. Web Frontend

Web frontend merupakan aplikasi berbasis web yang dibangun untuk user yang dapat digunakan untuk mengatur ID untuk perangkat yang digunakan pada infrastruktur dengan memanfaatkan protokol XMPP. User tidak langsung mengakses server melainkan menggunakan aplikasi web frontent untuk POST dan GET data ke server dengan menggunakan RestAPI. Gambar 6 merupakan *User Interface* dari Web Frontend yang telah dibuat.

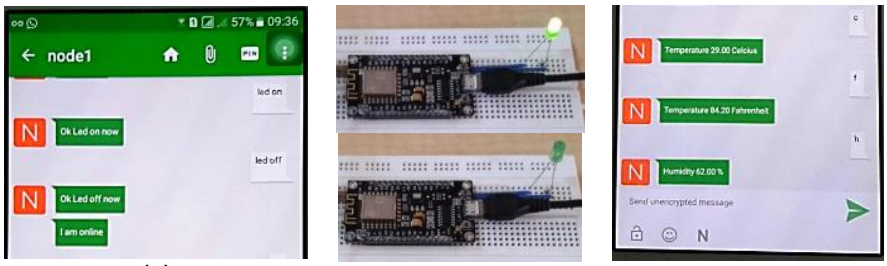


(a) (b)

Gambar 6. User Interface Web Frontend

3.2. Pengujian *Request-response* antar perangkat IOT

Pengujian *Request-response* dilakukan untuk menguji apakah perangkat IOT dapat saling berkomunikasi dan NodeMCU dapat memberikan *response* sesuai *request* dari perangkat lain yang terhubung menggunakan protokol XMPP. *Request* diberikan adalah mematikan dan menyalakan lampu LED serta melakukan sensing dengan DHT11. Gambar 8(a) merupakan *request* terhadap NodeMCU untuk menyalakan dan mematikan lampu LED, Gambar 8(b) merupakan kondisi lampu LED ketika kondisi menyala dan mati. Gambar 8(c) menunjukkan hasil sensing suhu dan kelembaban dengan menggunakan DHT11.



(a) (b) (c)

Gambar 7. Request-response terhadap NodeMCU

3.3. Pengujian Performansi protokol XMPP

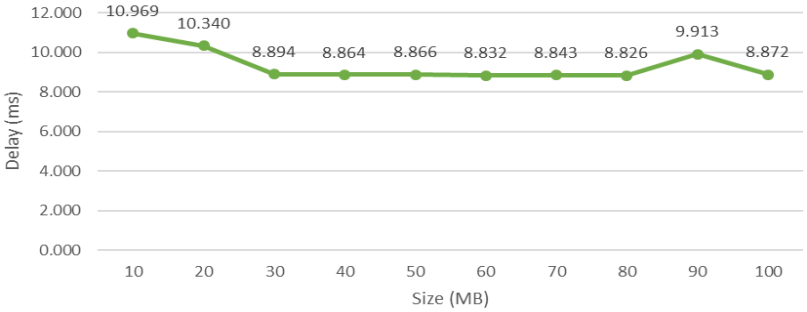
Pengujian ini bertujuan untuk menguji performansi dan efisiensi protokol XMPP dalam menangani transmisi data antar perangkat yang terdaftar pada jaringan protokol XMPP. Pengujian ini dilakukan dengan cara menganalisis *delay, jitter, throughput* yang dengan tools Wireshark serta load server pada saat transmisi antar perangkat IOT sedang berlangsung. Terdapat tiga skenario dalam mekanisme pengiriman data, yang pertama mengirim data dengan ukuran yang bervariasi dengan asumsi transfer rate yang konstan, kedua mengirim data dengan ukuran yang konstan dan transfer rate yang bervariasi, dan yang ketiga menguji load server saat kondisi *standby* dan saat transmisi data sedang berlangsung antar beberapa perangkat yang terhubung menggunakan protokol XMPP dan kemudian menganalis penggunaan CPU dan Memory pada server.

3.3.1. Pengujian variasi ukuran data

Ukuran data yang divariasikan antara 10MB hingga 100MB. Diasumsikan untuk transfer rate dalam keadaan konstan yaitu 100 Mbps dari Mikrotik RB750 ke modem, kecepatan *downlink* 77.08 Mbps, *uplink* 21.85 Mbps dari modem ke ISP, dan 100 Mbps dari perangkat menuju Mikrotik RB750. Pada server gateway *bandwidth* 3.22 Mbps, *downlink* 3.26 Mbps, dan *uplink* 5.0 Mbps menuju perangkat. Hasil pengujian *delay* pada kegiatan transmisi data menggunakan protokol XMPP pada Tabel 1 dibawah ini.

Tabel 1. Delay Transfer Data Skenario 1

Data	Size (MB)	Delay (ms)
1	10	10.969
2	20	10.340
3	30	8.894
4	40	8.864
5	50	8.866
6	60	8.832
7	70	8.843
8	80	8.826
9	90	9.913
10	100	8.872
Rata-rata		9.322

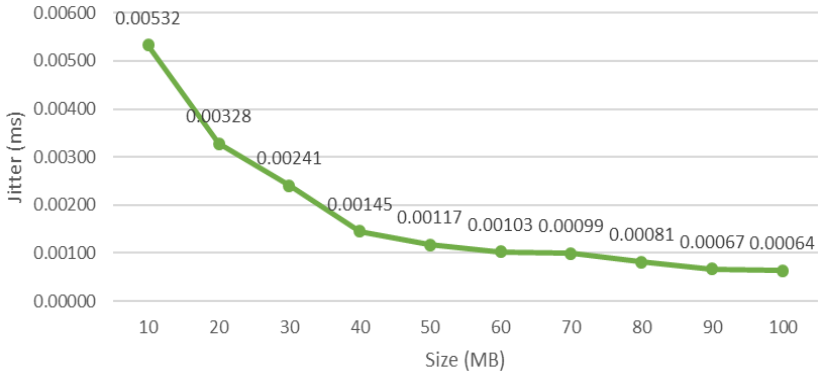


Gambar 9. Grafik Delay Transmisi Data Skenario 1

Dapat dilihat informasi pada Gambar 9 Jika diamati pada grafik, awalnya data pertama dan kedua memiliki grafik yang lebih tinggi daripada data yang lainnya, ini dikarenakan adanya mekanisme *handshaking* dimana koneksi pertama kali akan dibuat antara perangkat dan server sehingga waktu penundaan akan lebih tinggi, ketika koneksi telah terbuat maka waktu penundaan akan kembali stabil. Protokol XMPP bekerja dengan baik setelah koneksi dibuat karena pesan dikompresi menjadi berukuran kecil dan protokolnya sepenuhnya tersinkronisasi. Hasil pengujian *jitter* pada kegiatan transmisi data menggunakan protokol XMPP pada Tabel 2 berikut.

Tabel 2. Jitter Transfer Data Skenario 1

Data	Size (MB)	Jitter (ms)
1	10	0.00532
2	20	0.00328
3	30	0.00241
4	40	0.00145
5	50	0.00117
6	60	0.00103
7	70	0.00099
8	80	0.00081
9	90	0.00067
10	100	0.00064
Rata-rata		0.00178

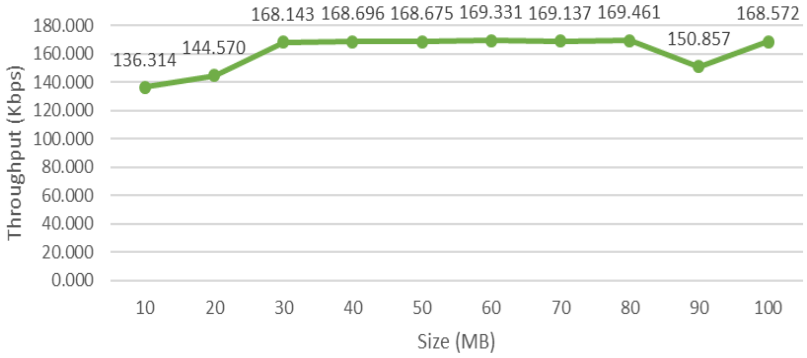


Gambar 10. Grafik Jitter Transmisi Data Skenario 1

Dapat dililhat informasi pada Gambar 10, adanya mekanisme *handshaking* pada awalnya membuat trafik agak sedikit terganggu sehingga nilai *jitter* pun tinggi kemudian ketika koneksi telah terbuat maka trafik akan membaik sehingga nilai *jitter* akan semakin menurun dan kinerja protokol XMPP akan membaik karena pesan yang dikompresi menjadi berukuran kecil dan protokolnya sepenuhnya tersinkronisasi. Walaupun ukuran data semakin besar tetapi koneksi telah terbuat sebelumnya sehingga protokol XMPP tersinkronisasi dengan baik saat koneksi kembali dibuat antar perangkat. Grafik menunjukkan penurunan nilai *jitter* saat transmisi data terus dilakukan yang berarti menurunkan nilai variasi *delay,* menunjukkan semakin baiknya protokol XMPP dalam mentransmisikan data saat transfer rate dalam keadaan konstan. Hasil pengujian *Throughput* pada kegiatan transmisi data menggunakan protokol XMPP pada Tabel 3 dibawah ini.

Tabel 3. Throughput Transfer Data Skenario 1

Data	Size (MB)	Throughput (Kbps)
1	10	136.314
2	20	144.570
3	30	168.143
4	40	168.696
5	50	168.675
6	60	169.331
7	70	169.137
8	80	169.461
9	90	150.857
10	100	168.572
Rata-rata		161.376



Gambar 81. Grafik Throughput Transmisi Data Skenario 1

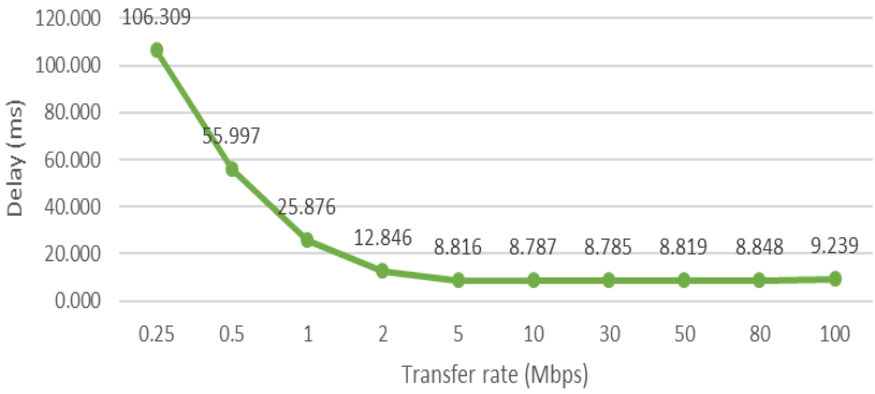
Dapat dililhat informasi pada Gambar 11. jika diamati pada grarfik, data pertama dan kedua memiliki grafik yang lebih rendah daripada data yang lainnya, ini dikarenakan adanya mekanisme *handshaking* dimana koneksi pertama kali akan dibuat antara perangkat dan server sehingga dengan proses tersebut kecepatan transfer menjadi lebih rendah dan ketika koneksi telah terbuat maka kecepatan transfer akan kembali stabil sehingga protokol XMPP bekerja dengan baik dan protokolnya sepenuhnya tersinkronisasi. Grafik *throughput* dengan grafik data *delay* menunjukkan pola kebalikan dimana jika nilai *delay* meningkat maka nilai *throughput* menurun begitu pun sebaliknya.

3.3.2. Pengujian variasi transfer rate

Transfer rate yang divariasikan yaitu pada mikrotik RB750 menuju modem dari 0.25 Mbps hingga 100 Mbps dengan ukuran file 20 MB. Karena variasi transfer rate saat transmisi data, maka akan berpengaruh terhadap *downlink* dan *uplink* dari pc/perangkat IOT menuju internet, seperti pada Tabel 4 berikut.

Tabel 4. Delay Transmisi Data Skenario 2

Data	Transfer rate (Mbps)	Delay (ms)
1	0.25	106.309
2	0.5	55.997
3	1	25.876
4	2	12.846
5	5	8.816
6	10	8.787
7	30	8.785
8	50	8.819
9	80	8.848
10	100	9.239
	Rata-rata	25.432

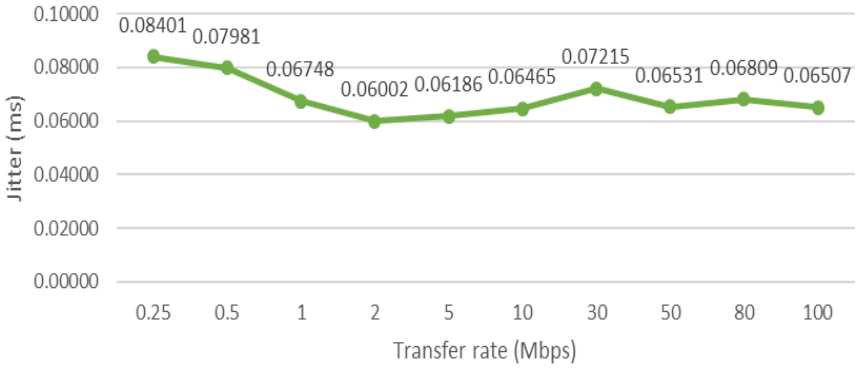


Gambar 92. Grafik Data Delay Transmisi Data Skenario 2

Dapat dilihat informasi pada Gambar 12. jika diamati dari grafik nilai *delay* akan berbanding terbalik dengan meningkatnya transfer rate. Hal ini dikarenakan transfer rate yang bervariasi sehingga trafik data perangkat juga menyesuaikan jalur yang dilalui serta berdampak pada waktu transmisi yang berlangsung. Jika diamati dari data tersebut grafik berhenti menurun pada transfer rate > 5 Mbps dan mulai menunjukkan grafik yang stabil. Hasil pengujian *jitter* pada kegiatan transmisi data menggunakan protokol XMPP pada Tabel 5 dibawah ini.

Tabel 5. Jitter Transmisi Data Skenario 2

Data	Transfer rate (Mbps)	Jitter (ms)
1	0.25	0.08401
2	0.5	0.07981
3	1	0.06748
4	2	0.06002
5	5	0.06186
6	10	0.06465
7	30	0.07215
8	50	0.06531
9	80	0.06809
10	100	0.06507
	Rata-rata	0.06885

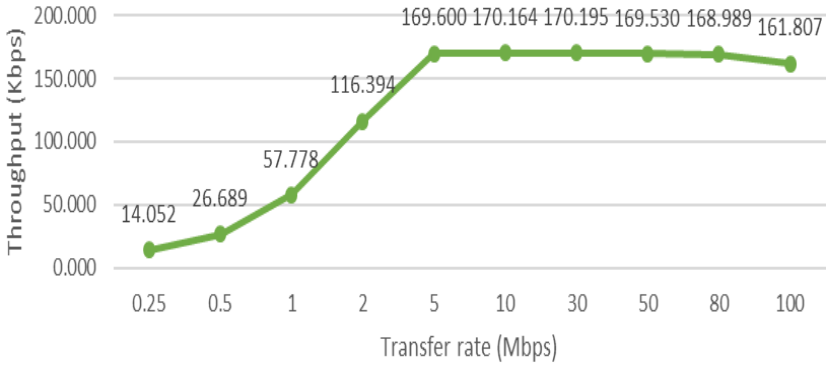


Gambar 13. Grafik Data Jitter Transmisi Data Skenario 2

Dapat dilihat informasi pada Gambar 13 jika diamati dari grafik tersebut nilai *jitter* cenderung stabil dengan penurunan dan kenaikan nilai yang tidak terlalu signifikan. Hal ini dikarenakan ukuran data yang konstan maka nilai selisih *delay* antar paket yang tertangkap pada Wireshark saat dianalisis juga konstan sehingga nilai variasi *delay* mengalami kenaikan dan penurunan yang tidak terlalu signifkan saat transmisi data antar perangkat berlangsung. Hasil pengujian *Throughput* pada kegiatan transmisi data menggunakan protokol XMPP pada Tabel 6 dibawah ini.

Tabel 6. Throughput Transmisi Data Skenario 2

Data	Transfer rate (Mbps)	Throughput (Kbps)
1	0.25	14.052
2	0.5	26.689
3	1	57.778
4	2	116.394
5	5	169.600
6	10	170.164
7	30	170.195
8	50	169.530
9	80	168.989
10	100	161.807
	Rata-rata	122.520



Gambar 14. Grafik Data Throughput Transmisi Data Skenario 2

Dapat dililhat informasi pada Gambar 14 jika diamati kenaikan mulai berhenti dari transfer rate > 5 Mbps dan mulai menunjukkan grafik yang stabil. Hal ini dikarenakan transfer rate yang bervariasi sehingga trafik data perangkat juga menyesuaikan jalur yang dilalui serta berdampak pada besarnya ukuran data yang mampu ditransmisikan pada jaringan. Jika diamati grafik data *throughput* dengan grafik data *delay* menunjukkan pola kebalikan dimana jika nilai *delay* meningkat maka nilai *throughput* menurun begitu pun sebaliknya

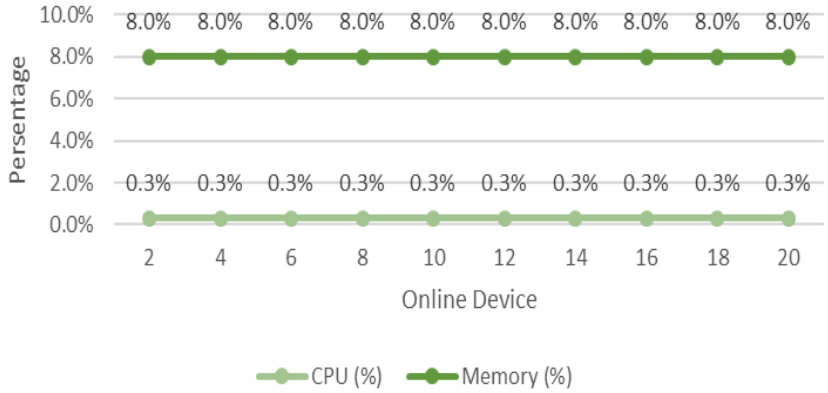
3.3.3. Pengujian Load server

Ukuran data yang transmisikan yaitu 20MB. Pengujian dilakukan dengan mengukur besar beban server saat beberapa perangkat terhubung sebagai *client* pada jaringan XMPP. Perangkat yang terhubung dan diuji yaitu 2 hingga 20 perangkat pada saat kondisi *standby* atau tidak ada akivitas dan saat kondisi dimana perangkat melakukan transmisi data antar perangkat yang terhubung pada server kemudian membandingkan hasil yang didapatkan. Saat transmisi data berlangsung, setiap perangkat dipasangkan dengan satu perangkat yang lainnya sehingga menghasilkan 1 koneksi setiap 2 perangkat yang terhubung. Diasumsikan untuk transfer rate dalam keadaan konstan yaitu 100 Mbps dari Mikrotik RB750 ke modem, kecepatan *downlink* 77.08 Mbps, *uplink* 21.85 Mbps dari modem ke ISP, dan 100 Mbps dari perangkat menuju Mikrotik RB750. Pada server gateway *bandwidth* 3.22 Mbps, *downlink* 3.26 Mbps, dan *uplink* 5.0 Mbps dari perangkat. Hasil pengujian load server saat perangkat yang terhubung dalam kondisi *standby* pada server menggunakan protokol XMPP pada Tabel 7 dibawah ini.

Dapat dililhat informasi pada Gambar 15 jika diamati dari grafik load server pada penggunaan CPU dan Memory dari awal hingga akhir grafik menunjukkan data yang konstan, tidak ada kenaikan ataupun penurunan pada grafik data. Nilai penggunaan CPU menunjukkan penggunaan 0.3 % dan Memory 8.0% pada server. Hasil pengujian load server saat transmisi data antar beberapa perangkat berlangsung pada server menggunakan protokol XMPP pada Tabel 8 dibawah ini.

Tabel 7. Load Server Kondisi Standby

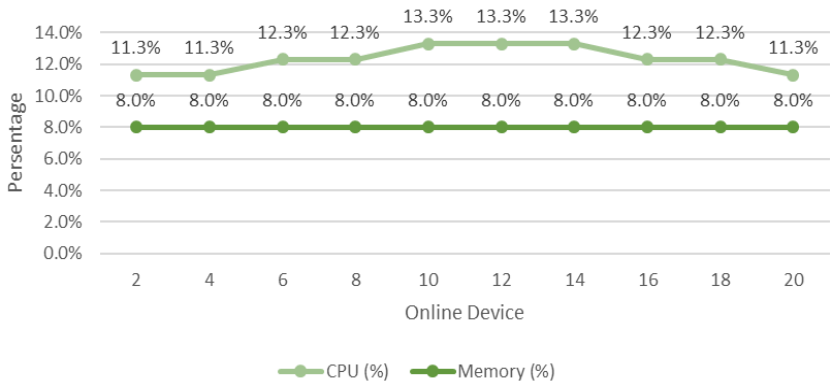
Data	Online Device	CPU (%)	Memory (%)
1	2	0.3	8.0
2	4	0.3	8.0
3	6	0.3	8.0
4	8	0.3	8.0
5	10	0.3	8.0
6	12	0.3	8.0
7	14	0.3	8.0
8	16	0.3	8.0
9	18	0.3	8.0
10	20	0.3	8.0
Rata-rata		0.3	8.0



Gambar 10 Grafik Load Server Saat Kondisi Perangkat Standby

Tabel 8. Load Server Saat Transmisi Data

Data	Online Device	CPU (%)	Memory (%)
1	2	11.3	8.0
2	4	11.3	8.0
3	6	12.3	8.0
4	8	12.3	8.0
5	10	13.3	8.0
6	12	13.3	8.0
7	14	13.3	8.0
8	16	12.3	8.0
9	18	12.3	8.0
10	20	11.3	8.0
	Rata-rata	12.3	8.0



Gambar 11 Grafik Load Server Saat Transmisi Data

Dapat dililhat informasi pada Gambar 16 jika diamati dari grafik load server saat transmisi data antara perangkat berlangsung penggunaan CPU mengalamai kenaikan dan penurunan yang cenderung stabil dan Memory dari awal hingga akhir grafik menunjukkan data yang konstan, tidak ada kenaikan ataupun penurunan pada penggunaan Memory. Nilai penggunaan CPU yang paling tertinggi 13.3% dan penggunaa CPU paling rendah 11.3% dengan rata-rata 12.3% sedangkan penggunaan Memory dari awal hingga akhir 8.0%.

4. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, penulis dapat menyimpulkan beberapa hal antara lain:

1. XMPP dapat digunakan sebagai protokol komunikasi karena dapat digunakan untuk berbagai macam *platform* dan *open-source* serta dokumentasi yang telah terdokumentasi secara baik sehingga dapat dengan mudah membangun sistem yang diinginkan.
2. Dengan adanya fitur Rest API dari Openfire, dapat digunakan untuk membangun aplikasi sesuai keinginan.
3. NodeMCU dan perangkat IoT dapat melakukan proses *response-request* dan melakukan intruksi sesuai dengan harapan pada perancangan.
4. Pengujian performansi protokol XMPP saat transmisi menunjukkan bahwa pada kondisi transfer rate yang baik, protokol XMPP mampu bekerja dengan baik karena koneksi yang dibuat sepenuhnya tersinkronisasi dengan baik setelah melewati mekanisme *handshake* dengan konsumsi CPU 0.3% dan memory 8.0% saat perangkat dalam kondisi standby dan rata-rata penggunaan CPU 12.3% dan memory 8.0% saat transmisi data antar perangkat berlangsung

Referensi

[1] M. Farooq, M. Waseem, S. Mazhar, A. Khairi and T. Kamal, "A Review on Internet of Things (IoT)," *International Journal of Computer Applications (0975 8887),* 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," *Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing,* pp. 60-69, 2012.

[3] Y. Wenbo, W. Quanyu and G. Zhenwei, "Smart Home Implementation Based on Internet and WiFi Technology," *Chinese Control Conference, CCC,* Vols. 2015-September, pp. 9072-9077, 2015.

[4] A. Junaidi, "Internet of Things , Sejarah , Teknologi Dan Penerapannya : Review Internet of Things , Sejarah , Teknologi dan Penerapannya : Review," *Jurnal Ilmiah Teknologi Informasi Terapan (JITTER),* vol. I, no. August 2015, pp. 62-66, 2016.

[5] A. Al-Fuqaha, M. Guizan, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials,* vol. 17, no. 4, pp. 2347-2376, 2015.

[6] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann and M. Ameling, "A service infrastructure for the Internet of Things based on XMPP," *2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2013,* no. March, pp. 385-388, 2013.

[7] P. Masek, J. Hosek, K. Zeman, M. Stusek, D. Kovac, P. Cika, J. Masek, S. Andreev and F. Kröpfl, "Implementation of True IoT Vision: Survey on Enabling Protocols and Hands-On Experience," *International Journal of Distributed Sensor Networks,* vol. 2016, 2016.

[8] M. B. Yassein, M. Q. Shatnawi and D. Al-zoubi, "Application Layer Protocols for the Internet of Things," in *2016 International Conference on Engineering MIS (ICEMIS)*, 2016.

[9] E. Zuliarso and H. Februariyanti, "Pemanfaatan Instant Messaging untuk Aplikasi Layanan Akademik," *Dinamik-Jurnal Teknologi Informasi,* vol. 18, no. 2, pp. 1-45, 2013.

[10] W. Yan, Q. Wang and Z. Gao, "Smart home implementation based on Internet and WiFi technology," in *Chinese Control Conference, CCC*, 2015.