Implementasi Protokol CoAP pada Smart Building berbasis OpenMTC

Haikal Rahmat Fadilah
Mahasiswa S1 Teknik Informatika
Telkom University
Bandung Indonesia
haikal@students.telkomuniversity.ac.id

Dr. Maman Abdurohman, S.T., M.T
Dosen Teknik Informatika
Telkom University
Bandung Indonesia
abdurohman@telkomuniversity.ac.id

Anton Herutomo ST, M.Eng
Dosen Teknik Informatika
Telkom University
Bandung Indonesia
Anton.herutomo@gmail.com

Abstrak— *Smart Building* merupakan teknologi dimana berbagai komponen dalam bangunan dalam saling berinteraksi. Pada *smart building* komunikasi menjadi aspek penting untuk menyampaikan informasi. Pada paper ini, dilakukan penelitian mengenai komunikasi pada *smart building*, dengan mengaplikasikan protokol CoAP dalam *platform middleware M2M* OpenMTC yang sebelumnya memiliki protokol standar, yaitu HTTP. Protokol CoAP berperan dalam komunikasi antara sensor atau *device application* dan GSCL OpenMTC. Pengujian kinerja dilakukan dengan pengiriman data sensor sebesar 10, 100, dan 1000 dan ditentukan dengan parameter analisis *delay*, *throughput*, dan *overhead* protokol. Jumlah tersebut dinilai mewakili banyaknya data sensor pada suatu bangunan. Hasil dari pengujian dan analisis menunjukan bahwa protokol CoAP memiliki *delay* yang lebih rendah, *throughput* yang lebih stabil ketika data sensor menuju 1000, serta protokol CoAP memiliki *overhead* sekitar 50% lebih rendah dibandingkan dengan protokol HTTP.

Kata kunci— Smart Building, M2M, OpenMTC, CoAP, HTTP, Kinerja.

Abstract— Smart Building is a technology where various components building interact each other. In smart building, communication is an importent aspect to deliver information. this paper focus on smart building communication by applying protocol CoAP in OpenMTC M2M middleware platform that previsously had standard protocol HTTP. CoAP protocol has located in communication between sensor or device application and GSCL OpenMTC. Performance testing use sensor data at 10, 100, and 1000 by sending sensor data from device application towards GSCL OpenMTC and were determined use parameter, that is delay, throughput, and overhead protocol. this value represent the amount of sensor data on a building. The results show that CoAP protocol has lower delay, more stable throughput when the sensor data reaches 1000, and overhead of CoAP protocol has approximately 50% lower compared with HTTP protocol.

Keywords— Smart Building, M2M, OpenMTC, CoAP, HTTP, performance.

I. PENDAHULUAN

Smart building merupakan perkembangan dari teknologi *internet of things*, dimana pada *smart building* dapat menghubungkan berbagai perangkat dalam bangunan atau sensor untuk saling berkomunikasi melalui jaringan. *Smart building* dikembangkan sebagai bagian dari kebutuhan manusia, yaitu untuk memudahkan pekerjaan yang dilakukan sehari-hari. Teknologi seperti ini diperkirakan akan menjadi tren dimasa yang akan datang.

Adanya konsep smart building didasari oleh perkembangan teknologi *Machine-to-Machine* (M2M). Karakteristik dari teknologi *Smart Building* adalah terdiri dari node dan *gateway*/*server* dimana dengan cara kerja sebuah node akan mengirimkan data ke *gateway* atau *server* untuk simpan dan diolah pada tahap selanjutnya.

Untuk menyimpan dan mengolah data yang didikirimkan oleh node tersebut diperlukan suatu *paltform middleware* sebagai *management application*, salah satunya *platform*-nya adalah OpenMTC. Selain itu, aspek penting dalam pengiriman data adalah protokol yang digunakan. Protokol komunikasi menentukan seberapa cepat dan seberapa efisien, baik *bandwidth* maupun energi untuk membawa data sampai ke tujuan. *Platform* OpenMTC pada awalnya menggunakan protokol standar yaitu HTTP. Namun mengingat diperlukannya peningkatan kualitas komunikasi pada *smart building*, maka dikembangkannya openMTC dengan protokol yang lebih moderen, dalam hal ini protokol CoAP. Protokol CoAP merupakan protokol yang memiliki kelebihan berupa *low overhead*, *multicast*, dan *simplicity* sehingga cocok untuk *constrained device*.

Penelitian ini akan difokuskan pada perbandingkan protokol yang digunakan untuk pengiriman data dari sebuah sensor menuju server GSCL OpenMTC, yaitu HTTP dan CoAP. Keduanya akan dilakukan pembandingan dengan parameter analisis, yaitu *delay*, *throughput*, dan protokol *overhead*.

II. TEORI DASAR

A. *OpenMTC Platform*

OpenMTC merupakan suatu implementasi dari M2M *middleware* yang bertujuan sebagai penyedia standar untuk *platform smart city, smart building* dan *machine-to-machine*. [3]

OpenMTC terdiri dari 2 bagian yaitu, pertama *Network Domain* dan kedua *Device*/*Gateway Domain*. Untuk menghubungkan keduanya terdapat 2 skenario, *Device* dapat langsung terhubung ke network domain melalui *access point*, atau *device* terhubung ke network domain melalu *gateway* sebagai *network proxy*. [3]

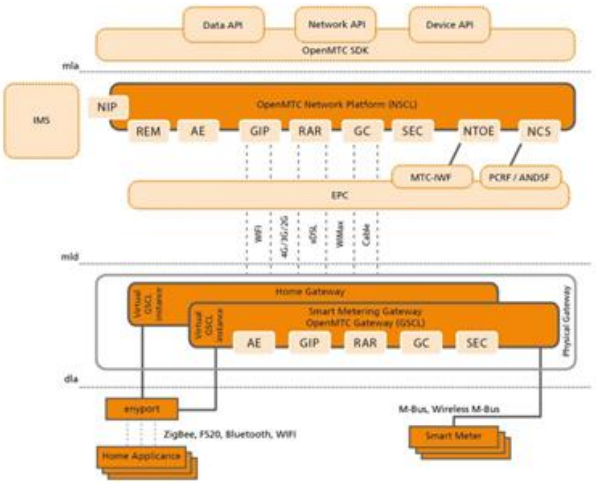
Kedua bagian tersebut dapat terhubung menggunakan M2M service *capability*. *Service capability* berguna saat melakukan komunikasi antar aplikasi M2M yang berbeda. Secara spesifik *Service Capability*, memiliki karakteristik sebagai berikut :

- Dapat mengoptimalkan pengembangan aplikasi dan menghubungkan jaringan yang tersembunyi dari aplikasi.
- Sebagai antarmuka antar satu atau beberapa *core network*.
- Memperlihatkan fungsionalias melalui *interfaces.*
- Menggunakan *core network functionalities*.

Service Capability (SC) layer terdiri dari 3 bagian, *Gateway Service Capability Layer* (GSCL) yang bekerja *pada Gateway, Device Service Capability Layer (*DSCL) yang bekerja pada *device* dan *Network Service Capability Layer* (NSCL) yang bekerja pada *network.*[3]

1. *Generic Communication* (xGC) – DGC, GGC, NGC
2. *Application Enablement* (xAE) – DAE, GAE, NAE
3. *Reachability*, *Addressing* and *Repository* (xRAR) – DRAR, GRAR, NRAR
4. *Remote Entity Management* (xREM) – DREM, GREM, NREM
5. *Interworking Proxy* (xIP) – DIP, GIP, NIP
6. *Security Capability* (SEC) – DSEC, GSEC, NSEC
7. *Network Communication Selection* (NCS)
8. *Network Telco Operator Exposure* (NTOE)

Beberapa contoh penggunaan OpenMTC dalam komunikasi M2M diantaranya untuk memonitor sinyal *electrocardiagram* yang dikirimkan oleh aktivitas *electric* jantung[10]. Selain itu juga OpenMTC telah digunakan untuk *mobile tracking system* pada kendaraan dimana pada system ini akan mengirimkan berbagai informasi posisi kendaraan seperti kecepatan, suhu kabin, dan jumlah penumpang [11].



Gambar 1. Arsitektur OpenMTC

B. *Constrained Application Protocol (CoAP)*

Constrained Application Protocol (CoAP) merupakan protokol komunikasi yang digunakan pada *constrain devices* atau perangkat yang memiliki keterbatasan, seperti energi, RAM dan sebagainya. CoAP memiliki kelebihan yaitu menggunakan *power* yang rendah dalam pengoprasiannya. CoAP umumnya digunakan untuk perangkat yang memerlukan energi yang rendah seperti sensor, switch atau komponen lain yang memerlukan pengendalian. [4]

Pada dasarnya CoAP hampir sama dengan HTTP, CoAP mengadopsi berbagai *pattern* dari HTTP seperti resource abstraction, URIs, RESTful interactions dan extended header. Akan tetapi CoAP menggunakan *binary* dalam representasinya dan dalam transimisinya CoAP menggunakan protokol UDP sehingga CoAP bersifat *multicast*, yaitu dapat melakukan komunikasi *one*-to-*many*. [4]

Cara kerja protocol CoAP secara keseluruhan hampir sama dengan HTTP. Pada protocol CoAP membagi tipe message menjadi 4, yaitu *Confirmable*(CON), *Non-Confirmable*(NON), *Acknowledgement*(ACK), dan Reset(RST), dimana *method code* dan *response* code sudah termasuk didalamnya[4].

Messaging model pada CoAP memiliki dasar dimana pertukaran message melalui UDP. Format *message* CoAP menggunakan short fixed-length binary hearder(4 bytes) yang diikuti *compact binary* options dan payloadnya. Setiap pesan berisi message ID yang mendeteksi duplikasi dan untuk reliability. Message ID berukuran 16 bit yang dapat menampung hinga 250 *message* persecondnya. *Message* CoAP di *encode* dalam bentuk *binary*[4].

Message yang dikirim oleh CoAP memiliki dua bentuk, yaitu *Reliability* dan *non realibility*. *Reliability* digunakan pada *Confirmable message* untuk memastikan

pesan sampai kepada penerima. *Confirmable* message akan terus dikirim ulang jika terjadi timeout hingga penerima mengirim ACK dengan *message ID* yang sama. Selain itu, terdapat *message* yang tidak memerlukan reliable transmisi atau disebut non *realibility* (umumnya message untuk sensor). *Message* akan dikirim dengan *Non-confirmable* message (NON) dimana tidak memiliki ACK, tetapi tetap memiliki message ID. [4].

Request/response ada didalam CoAP *message* bersama *method code* atau *response code*. *Request* dibawa oleh Comfirmable (CON) atau Non- *Confirmable message* sedangkan response dibawa oleh *Acknowledge message* (ACK) yang dikirim kembali oleh penerima. Method code yang digunakan oleh CoAP sama seperti HTTP, yaitu GET, PUT, POST dan DELETE[4].

C. Smart Building

Smart building merupakan kumpulan dari instalasi dan integrasi berbagai sistem teknologi bangunan. Sistem dalam bangunan dapat berupa, otomatisasi, keamanan, telekomunikasi, *user systems*, dan sistem manajemen fasilitas. *Smart building* sebagai sebuah sistem intelegensi yang membantu menjalankan berbagai fasilitas bangunan secara efisien. Sistem ini terdiri dari sensor, software, controller, dan connection. [8]

Asal mula adanya *smart building* merupakan adanya permintaan dari manusia untuk mengetahui performansi dari bangunan yang mereka miliki, sehingga mereka dapat melakukan pengurangan biaya operasi, mencapai tujuan pekerjaan sehari-hari, memberikan keamanan serta kenyamanan dan meningkatkan produktifitas.[8] Salah satu contoh penerapan *smart building* seperti penggunaan sensor *lighting* dan sensor *temperature* untuk efisiensi energi.

Dalam *smart building* sensor *lighting* dapat berguna untuk mendeteksi adanya cahaya pada suatu ruang bangunan, dimana jika terdapat cahaya yang mencukupi, sensor *lighting* tersebut akan menonaktifkan atau mengurangi penggunaan cahaya lampu dalam ruangan secara otomatis dan sebaliknya. Sedangkan sensor *temperature* berguna untuk mendeteksi besarnya suhu yang didapat dan dikirimkan ke server untuk pengolahan data selanjutnya[9].

D. Quality of Service

1. *Overhead* Protokol

Protokol *overhead* merupakan alokasi "non data" dari suata transmisi paket. *Overhead* dapat dikatakan informasi yang harus dikirim bersamaan dengan data utama. Informasi tersebut berperan dalam menyampaikan data utama ke tujuan. Berikut perhitungan untuk mengukur protokol *overhead* [2] :

$$Overhead = \frac{total\ header}{(total\ header + data)}$$

2. Throughput

Throughput merupakan jumlah bit yang terkirim dari suatu perangkat ke perangkat lain pada waktu tertentu. *Throughput* dapat dipengaruhi oleh beberapa faktor, diantaranya tipe data yang dikirim, topologi jaringan, banyaknya pengguna jaringan, kondisi fisik medium atau spesifikasi perangkat yang digunakan. Berikut perhitungan untuk mengukur *throughput* [6]:

$$Throughput = \frac{\text{jumlah ukuran paket yang diterima}}{\text{waktu antara paket pertama dan terakhir}}$$

Berdasarkan[7] ETSI LTN besarnya *througnput* untuk M2M < 50 kbps

3. Delay

Delay merupakan waktu tunda yang diakibatkan proses transmisi dari suatu perangkat ke perangkat lain. Delay yang digunakan adalah *one way delay* Berikut perhitungan untuk mengukur *delay* [6]:

$$Delay = \frac{\sum \text{waktu antara paket dikirim dan paket diterima}}{\sum \text{jumlah paket terkirim}}$$

Menurut[5] Eurocom Institut dalam publikasi LOLA Project besarnya delay M2M < 1290 ms.

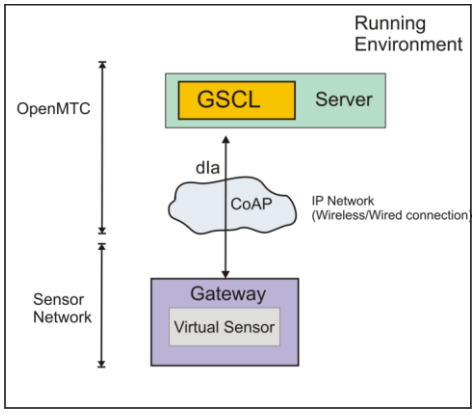
III. PERANCANGAN SISTEM

A. Gambaran Umum Sistem

Rancangan sistem yang dibangun terdiri dari dua bagian, Dalam penelitian ini, sistem yang dibangun merupakan sistem komunikasi *smart building* yang menghubungkan antara sensor virtual dan sebuah server berbasis *platform* OpenMTC dengan penambahan protokol komunikasi CoAP sebagai protocol komunikasinya.

Sistem terbagi menjadi dua bagian, yaitu OpenMTC dan bagian Sensor network. Pada bagian OpenMTC, terdiri dari aplikasi dan *service capability*. Aplikasi berfungsi untuk mengekspos service dan data yang terdapat dalam OpenMTC. Sedangkan *service capability* yang menyediakan berbagai fungsi komunikasi. *Service capability* OpenMTC bekerja pada dua bagian, yaitu *network* dengan layanan *Network Service Capability Layer* (NSCL) dan *gateway* dengan layanan *Gateway Service Capability Layer* (GSCL). Secara umum, NSCL pada OpenMTC sebagai penghimpun satu atau lebih gateway sensor, sedangkan GSCL pada OpenMTC berperan sebagai penghimpun data yang dihasilkan oleh sensor. Dalam penelitian ini hanya digunakannya GSCL yang

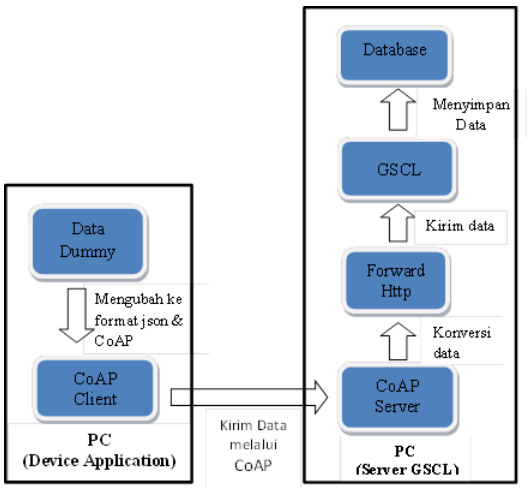
diimplementasikan protokol CoAP untuk komunikasi antara GSCL dan sensor virtual. Bentuk implementasi protokol CoAP dengan membuat aplikasi proxy pada salah satu *service capability*, yaitu GIP (*Gateway Interworking Proxy*) yang mempunyai peran dalam komunikasi GSCL. Berikut model sistem yang akan diimplementasikan[3]:



Gambar 2. Gambaran Sistem

Aplikasi proxy yang dibuat merupakan modul tambahan yang digunakan sebagai perantara antar dua protokol. Aplikasi tersebut menggunakan *package* CoAP yang dikembangkan oleh MIT[1] dengan fungsionalitas hanya berupa pengiriman dari suatu node ke server dan format message yang digunakan adalah *Confirmable Message*. Proxy bekerja dengan meneruskan paket dari protokol CoAP ke protokol HTTP. Hal ini dikarenakan OpenMTC memiliki struktur modul yang hanya bisa diterima dalam bentuk HTTP

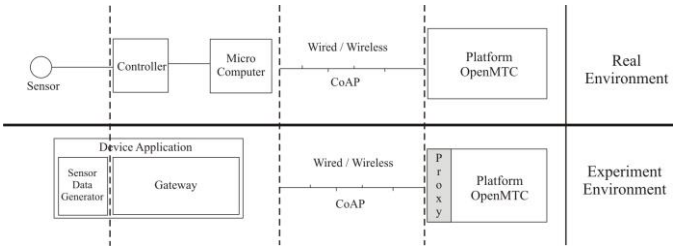
.



Gambar 3. Aliran proses sistem

B. Lingkungan Eksperimen vs real

Berikut merupakan pembandingan sistem yang dibangun pada penelitian dan sistem yang ada pada dunia nyata.



Gambar 4. Perbandingan lingkungan eksperimen vs real

C. Skenario Pengujian

Penelitian ini menggunakan infrastruktur jaringan LAN dan jaringan WLAN. Pengiriman data sensor dilakukan dengan dua infrastruktur tersebut sebagai ilustrasi penggunaan infrastruktur jaringan pada *smart building* yang sebenarnya.

Pada proses simulasi dan pengujian, pengambilan data dilakukan dengan skenario yang berbeda, dimana skenario dibagi menjadi tiga macam dengan jumlah sensor berbeda, hal ini bertujuan untuk mengilustrasikan kemampuan protokol dengan jumlah sensor yang berbeda. Berikut skenarionya antara lain :

Tabel 1. Skenario Pengujian via LAN dan WLAN

No.	Protokol	Jumlah Sensor	Parameter
1	HTTP	10	Delay Throughput Overhead
		100	
		1000	
2	CoAP	10	Delay Throughput Overhead
		100	
		1000	

Besarnya jumlah sensor tersebut digunakan untuk mewakili jumlah data sensor pada *smart building*, dimana 10 data sensor mewakili lingkup suatu ruangan, 100 data sensor memiliki lingkup perlantai, dan 1000 data sensor melingkupi seluruh bangunan *smart building*.

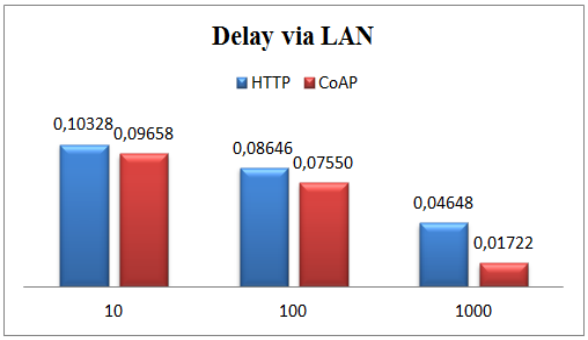
IV. HASIL DAN PEMBAHASAN

Pada bab ini memuat hasil pengujian yang dilakukan sesuai dengan skenario uji yang telah ditentukan sebelumnya. Serta analisis dari setiap hasil pengujian. Pengujian bertujuan untuk menjawab hasil dari rumusan masalah dan hipotesa yang ada bab sebelumnya. Berikut merupakan hasil pengujian dan analisis yang ditekankan pada parameter *delay*, *throughput*, dan *overhead*, antara lain :

A. Pengujian Delay

Tabel 2. Perbandingan Delay via Jaringan LAN

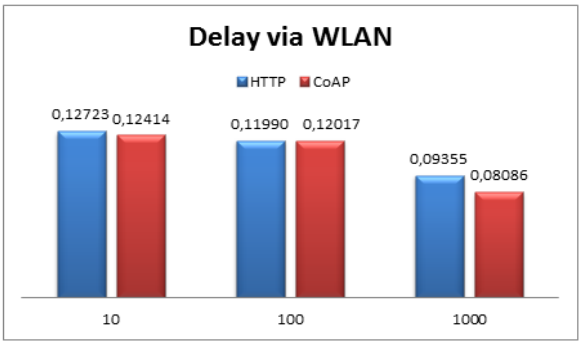
Jumlah sensor	Rata-rata Delay (s)	
	HTTP	CoAP
10	0,103278810	0,096580400
100	0,086456199	0,075501215
1000	0,046476157	0,017224287



Gambar 5. Grafik Delay via LAN

Tabel 3. Perbandingan Delay via Jaringan WLAN

Jumlah sensor	Rata-rata Delay(s)	
	HTTP	CoAP
10	0,127231670	0,124135000
100	0,119899760	0,120171240
1000	0,093545271	0,080862391



Gambar 6. Grafik Delay via WLAN

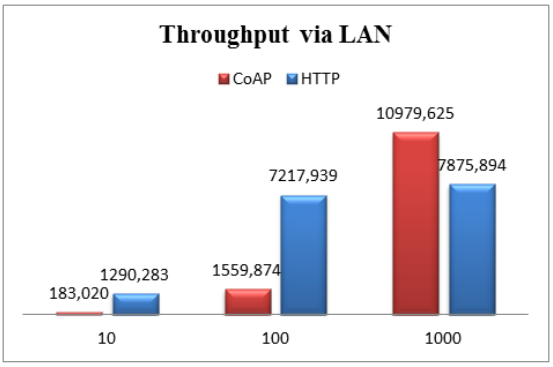
Berdasarkan hasil pengujian diatas, untuk *one way delay* antara waktu saat paket dikirim hingga paket diterima oleh server, protokol CoAP memiliki *one way delay* yang rata-rata lebih rendah jika dibandingkan dengan protokol HTTP, baik pada jaringan LAN maupun jaringan WLAN dikarenakan paket CoAP yang bersifat *simplicity* dan *connectionless*, yaitu tidak adanya mekanisme *three-way handshake*, *sequencing* dan *timers*. Pada hakikatnya paket CoAP menyerupai paket UDP dimana data dikirim secara langsung tanpa menghiraukan hal-hal tersebut. Berbeda dengan protokol HTTP dengan bentuk paket TCP yang memiliki karakteristik sebaliknya. Semakin kecilnya *delay* ketika sensor semakin banyak dipengaruhi oleh *throughput* yang semakin besar.

Untuk jenis LAN atau WLAN, terlihat bahwa jaringan LAN memiliki *delay* yang lebih stabil, dikarenakan pada jaringan WLAN dapat dipengaruhi oleh faktor kekuatan sinyal dan jumlah pengguna yang terhubung dalam satu *access point.*

B. Pengujian Throughput

Tabel 4. Perbandingan Throughput via Jaringan LAN

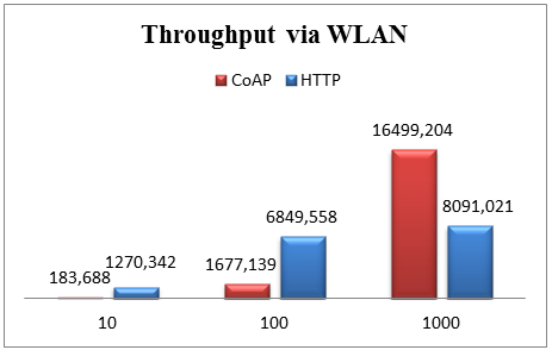
Jumlah sensor	Rata-rata Throughput / (bytes/sec)	
	HTTP	CoAP
10	1290,283	183,020
100	7217,939	1559,874
1000	7875,894	10979,625



Gambar 7. Grafik Throughput via LAN

Tabel 5. Perbandingan Throughput via Jaringan WLAN

Jumlah sensor	Rata-rata Throughput (bytes/sec)	
	HTTP	CoAP
10	1270,342	181,781
100	6849,558	1677,139
1000	8091,021	16499,204



Gambar 8. Grafik Throughput via WLAN

Berdasarkan hasil pengujian *throughput* diatas, terlihat bahwa dalam percobaan tersebut baik LAN maupun WLAN, setiap data yang didapat selalu mengalami perubahan, disebabkan oleh faktor-faktor yang telah disebutkan sebelumnya.

Pada grafik 4-3 dan 4-4, didapat data bahwa data sensor 10 dan 100, protokol CoAP memiliki *throughput* yang lebih rendah dibandingkan dengan protokol HTTP karena

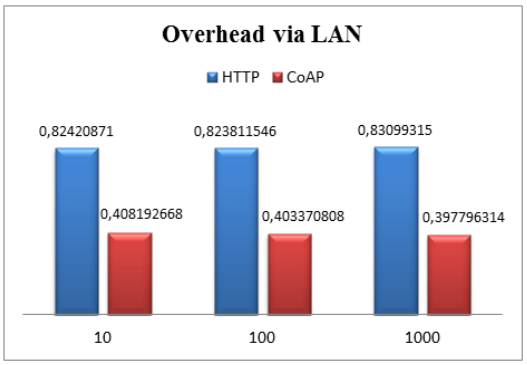
jumlah data yang dikirimkan oleh CoAP jauh lebih sedikit. Hal tersebut dipengaruhi bentuk paket menyerupai UDP yang dimiliki oleh CoAP sehingga paket yang dikirim lebih sedikit, berbanding terbalik dengan HTTP yang memiliki bentuk paket TCP, dimana adanya pengiriman SYN ACK sebelum melakukan pengiriman data, sehingga paket yang dikirimkan lebih banyak.

Namun, ketika pengiriman data sensor mencapai mencapai 1000 data, protokol HTTP terlihat adanya tanda-tanda ingin mencapai titik maksimal dengan terjadinya penurunan *throughput* seiring dengan bertambahnya jumlah data. Hal ini terlihat dengan *throughput* protokol CoAP yang lebih besar dibanding HTTP. Penurunan diakibatkan trafik jaringan yang tinggi karena semakin banyaknya data di jaringan. Pada protokol CoAP hingga mencapai 1000 data sensor belum terlihat tanda-tanda telah mencapai titik maksimal. Peristiwa tersebut terjadi pada jaringan LAN maupun WLAN dengan *throughput* yang tidak berbeda jauh.

C. Pengujian Overhead

Tabel 6. Perbandingan Overhead via Jaringan LAN

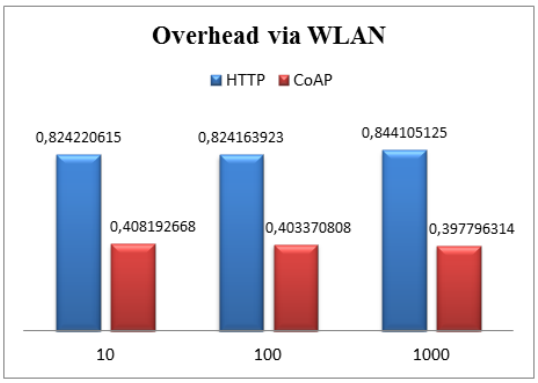
Jumlah sensor	Rata-rata overhead	
	HTTP	CoAP
10	0,824208710	0,408192668
100	0,823811546	0,403370808
1000	0,83099315	0,397796314



Gambar 9. Grafik Overhead via LAN

Tabel 7. Perbandingan Overhead via Jaringan WLAN

Jumlah sensor	Rata-rata overhead	
	HTTP	CoAP
10	0,824220615	0,408192668
100	0,824163923	0,403370808
1000	0,844105125	0,397796314



Gambar 10. Grafik Overhead via WLAN

Berdasarkan hasil pengujian, secara garis besar paket yang dikirim menggunakan protocol CoAP memiliki *overhead* sekitar 50% lebih rendah dibandingkan dengan protokol HTTP. Hal ini terbukti bahwa sifat dari CoAP yaitu *low overhead* dan *lightweight*, disebabkan protokol CoAP menggunakan format menyerupai UDP dalam pengirimannya dengan ukuran *header* sebesar 8 bytes. UDP merupakan protokol yang bersifat *connectionless dan simplicity*, yaitu tidak melakukan negosiasi seperti halnya HTTP sesaat sebelum pengiriman data, tidak ada *sequencing* dan *timers*, sehingga paket yang dibawa lebih ringan.

Selain itu juga pada CoAP dan HTTP dari data yang didapat menunjukan bahwa semakin besar data yang dikirim, semakin kecil *overhead* yang dihasilkan. Penurunan tersebut dipengaruhi oleh semakin besarnya data yang dibawa, namun besar protokol *overhead* perdatanya tetap. Jika dibandingan dari sisi media transmisi, baik LAN maupun WLAN dapat dilihat bahwa keduanya tidak memiliki pengaruh.

V. PENUTUP

A. Kesimpulan

Berdasarkan pengujian dan analisis kinerja protokol CoAP dan protokol HTTP pada *Smart Building* berbasis platform OpenMTC, maka diperoleh kesimpulan :

1. Pengujian protokol CoAP dengan parameter *one way delay* saat paket terkirim hingga paket diterima, disimpulkan protokol CoAP memiliki *one way delay* yang lebih baik dibandingkan HTTP, dikarenakan paket CoAP yang bersifat *lightweight*.
2. Pengujian protokol CoAP dengan parameter *throughput* dihasilkan bahwa CoAP memiliki *throughput* yang lebih baik dibandingkan protokol HTTP, dimana protokol CoAP mampu mengirimkan data sensor hingga 1000 tanpa adanya penurunan *throughput*.
3. Pengujian protokol CoAP dengan parameter *overhead*, disimpulkan bahwa protokol CoAP memiliki *overhead* protokol sekitar 50% lebih rendah dibandingkan protokol HTTP. Hal ini dikarenakan protokol CoAP menggunakan format

menyerupai UDP dengan ukuran *header* 8 *bytes* dan tidak ada mekanisme *three-way handshake*.

4. Berdasarkan hasil pengujian tersebut, penggunaan proxy untuk protokol CoAP dengan fungsionalitas hanya berupa pengiriman dan dengan bentuk *message* yang *confirmable* memiliki hasil yang lebih baik pada *smart building* dengan jumlah data sensor 10,100, dan 1000 dibandingkan dengan protokol HTTP.

B. Saran

Berdasarkan penelitian Tugas Akhir ini, penulis memiliki saran untuk pengembangan selanjutnya, yaitu :

1. Melakukan implementasi protokol CoAP pada OpenMTC secara murni atau perubahan keseluruhan OpenMTC dengan mengaplikasikan berbagai fungsionalitas lain untuk memberikan hasil yang lebih efektif.
2. Mengimplementasikan protokol CoAP tidak hanya pada bagian *Gateway* Sensor – GSCL. Namun juga pada GSCL – NSCL OpenMTC.
3. Melakukan implementasi dan pengujian dengan topologi jaringan yang lebih kompleks.

REFERENSI

[1] Collina, Matteo. https://www.npmjs.com/package/coap. Diakses pada Desember 2014. MIT Lincense.

[2] Dunn, Brian P. Overhead In Communication Systems As The Cost of Constraints. 2010. Indiana : University of Notre Dame.

[3] F. FOKUS, "OpenMTC Platform A Generic M2M Communication Platform," 2012.

[4] I Shelby and Hartke. 2014. *RFC 7252 The Constrained Application Protocol (CoAP)*. German : Universitaet bremen TZI.

[5] Knopp, Raymond. Latency Requirement in M2M Application Scenarios. Eropa : Institut Eurocom.

[6] Yuvandra, M.Zulfin. Analisis kinerja trafik video chatting pada Sistem client-client dengan aplikasi wireshark. Medan : Universitas Sumatra.

[7] ETSI. Low Throughput Networks (LTN); Use Cases for Low Throughput Networks. ETSI GS LTN 001 V1.1.1 (2014-09).

[8] Hoffmann, Terry. 2009. *Smart Buildings*. Johnson Control, Inc.

[9] MISTRAL. Building intelligence with Smart Building Automation System (SBAS). Mistral Solutions Pvt Ltd.

[10] Suryani, V, dkk. 2013. *Electrocardiagram monitoring on OpenMTC platform.* Bandung: Telkom University.

[11] Abdurohman, M. dkk. 2013. Mobile tracking system using OpenMTC platform based on event driven method. Bandung: Telkom University.