RL-001

スマートシティのための MQTT プラットフォームの検証
An Evaluation of a Platform Using MQTT for Buildings and Smart Cities

粕谷 貴司† 近藤 正芳† 茂手木 直也† 松岡 康友† 矢野 雅†
秋山 貴紀‡ 境野 哲‡ 貞田 洋明‡ 堀越 崇‡ 畠山 英之‡
Takashi Kasuya Masayoshi Kondo Naoya Motegi Yasutomo Matsuoka Tadashi Yano
Takanori Akiyama Akira Sakaino Hiroaki Sadata Takashi Horikoshi Hideyuki Hatakeyama

1. はじめに

京都議定書や省エネ法の改定，東日本大震災による電力不足などにより，節電および温室効果ガス削減が喫緊の課題となっている．一方で，それらを解決し，更なる付加価値を提供するスマートシティを実現する取り組みが海外，国内の各所で行われている．

我々はスマートシティを実現する技術の 1 つとして注目を集めている MQTT[1]（MQ Telemetry Transport）を建築設備システムへ適用した．また，建築設備への適用を目的に仕様策定が進められている IEEE1888[2] と MQTT の連携をクラウド環境において実現し，ビル設備・スマートシティへの適用に充分なスケーラビリティ，堅牢性，相互接続性を持つことを検証した．本稿ではそれらの取り組みと成果について述べる．

2. ビルコミュニケーションシステム®

竹中工務店では「ビルコミュニケーションシステム®」（以下「ビルコミ」）という名称で，ビルの入居者や管理者に空調・照明の遠隔制御やエネルギーの見える化など，付加価値の高いサービス提供を行うシステム，およびプラットフォームの開発・検証を行っている[3]．本節ではビルコミの概要と，本検証においてビルコミの基幹技術として採用した MQTT, OSGi[4], IEEE1888 について述べる．

2.1 背景

従来，建物内の設備システムは，閉鎖的なネットワーク構成や独自プロトコルの採用が多かった．インターネット技術の普及により，IP による設備ネットワークの統合が進み，米国においては 1990 年中頃から，設備制御を目的とする BACnet[5]や Lonworks[6]といったオープンプロトコルの仕様策定が行われてきた．日本においても，独自仕様を加えながらそれらの標準化が進められ，現在では一般化した技術となっている．

また，ネットワーク回線の低廉化，クラウドコンピューティング技術の発達によって，ビルのデータを安全・安価に送ることが可能になった．それにより，エネルギー見える化システム，空調機などの遠隔制御システムなどがクラウド環境から多く提供されるようになってきた．

一方，東日本大震災による深刻な電力不足は，BEMS（Building Energy Management System）を導入した事業者を束ねて面的な節電を実現する，BEMS アグリゲータなどの取り組みを後押しすることとなった．また，米国では一般化している OpenADR[7]などのデマンドレスポンス技術も，日本での導入に向け実証が進められている．

加えて，センサネットワークの発展は M2M (Machine to Machine）通信やビッグデータ解析につながってきた．建物内のセンサデータ，施設内のワーカー等の位置情報を使って行動パターンの解析を行った事例[8]も増えてきている（図 1）．

しかしながら，クラウドのアーキテクチャや M2M，ビッグデータ解析などの技術は，閉鎖的なネットワークを志向している建築設備由来の BACnet などと相性が悪い．それらと連携するためには，建物側に適切にゲートウェイを設けて通信・制御の手段を確立する必要がある．

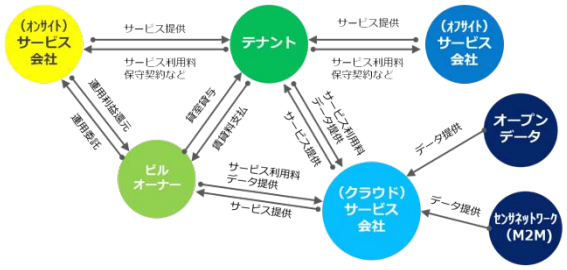

図 1　ビルにおけるサービス提供

2.2 システム概要

前述の背景を受けて，我々はクラウド環境を用いて，省エネや入居者の満足度向上に寄与するシステム開発・検証を行ってきた．システムの与件としては，リアルタイム性，スケーラビリティ，堅牢性，ユーザビリティが重要と考えており，そのための技術として，MQTT，OSGi を採用した．また，BACnet や IEEE1888，OpenADR といったプロトコルとのゲートウェイを開発し，相互運用性を確保した（図 2）．竹中工務店ではこれらのアーキテクチャや制御技術をビルコミと呼んでいる．

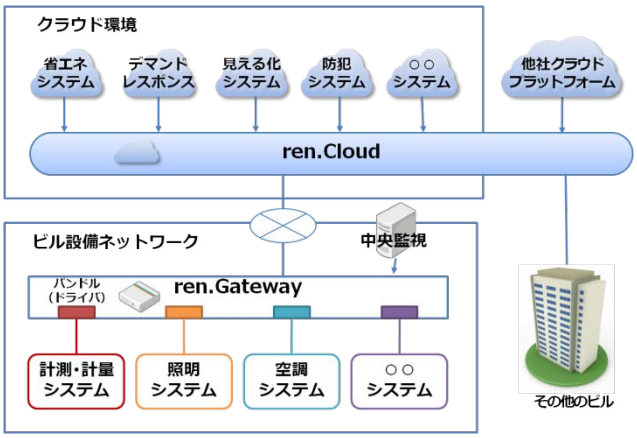

図 2　ビルコミュニケーションシステム®

図 2 の ren.Cloud は MQTT のメッセージを扱うクラウド型のプラットフォームであり，本検証では NTT コミュニケーションズの Biz ホスティング Cloudn [9]を利用している．また，ネットワーク回線としては，strongSwan[10]を用いた VPN を採用した．ren.Gateway は BACnet などの様々なローカルプロトコルと MQTT とのゲートウェイであり，OSGi バンドルとして実装することが可能である．

2.3 MQTT

我々は様々なアプリケーションをクラウドから提供するための仕組みとして，Publish /Subscribe 型のメッセージングプロトコルに注目した．JMS[11], STOMP[12], MQTT などがあるが，言語非依存であること，軽量であること，QoS の指定ができること，オープンかつライブラリが充実していることなどの理由から MQTT を採用した．

MQTT は IBM 社によって仕様策定が進められたプロトコルで，元々は橋梁の監視を行うセンサネットワークへの適用のために開発されていたが，大量のデータを効率的に扱えるために，様々な分野へ利用が広がっている．

MQTT の通信は，メッセージの発信側(Publisher)と受信側(Subscriber) が MQTT Broker と呼ばれるサーバを経由して行われる．また，建物内のセンサやアクチュエータなどには Topic と呼ばれる名称が割り当てられる．Publisher は MQTT Broker に対して，特定の Topic に割り当てられたセンサなどの情報を Publish し，Subscriber はシステムが取得したい情報を Subscribe することでリアルタイムの情報取得が可能になる．（図 3）

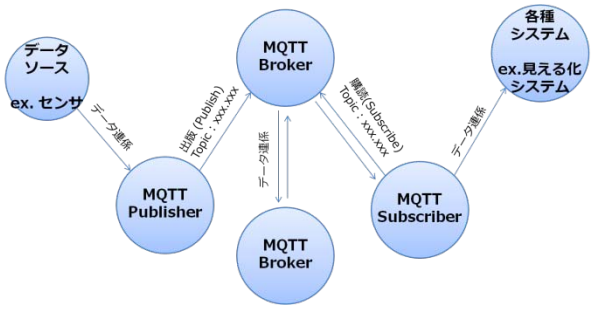

図 3 MQTT による通信

2.4 OSGi

OSGi は遠隔から管理できる Java ベースのサービスプラットフォームで，バンドルと呼ばれるコンポーネントを動作・連携させることでサービス提供を行うことができる．用途としては携帯電話や Eclipse などの総合開発環境，家庭用の情報機器を連携させるためのホームゲートウェイなどがある．多くのオープンソースでの実装が存在しており，遠隔から管理が簡便である特徴から，M2M への適用も進められてきている．Eclipse Kura Project[13]は OSGi をベースとした M2M における取り組みの 1 つであり，基幹のメッセージングサービスとして MQTT を採用している．

2.5 IEEE1888

IEEE（米国電気電子学会）が 2011 年に策定した国際的な通信規格の一つで「次世代の監視制御のための通信規格」として開発された．①コミュニティーの電力管理・設備管理 ②IT システムとの連携 などを得意とする ．東京大学の GUTP（Green University of Tokyo）が中心になって規格化している．実装は HTML＋XML であり，MQTT に比べてオーバーヘッドが大きいのが課題であるが，日本においては GUTP が牽引役となっており，各社の設備機器への標準実装も進んでいる．

本検証において，我々は MQTT と IEEE1888 のゲートウェイを作成し，エネルギー消費量の見える化システムと接続することで，その性能評価と課題の抽出を行った．詳細については，4.3 節で述べる．

3. 実証実験の環境

本検証では，ビルコミを使って竹中工務店の東京本店，技術研究所の建築設備の計量データを 1 分ごとに取得し，MQTT に変換して Biz ホスティング Cloudn へ送信している（図 4）．

計量データの取得は，設備ネットワークにある各種コントローラに，ビルコミの OSGi のバンドルが周期的に問い合わせることで行われ，ren.Gateway で MQTT に変換されている．なお，計量データの種類は電力量，温度，湿度，発電量など併せて 4000 ポイントほどである．また，クラウド環境で計量データを受け取る MQTT Broker に対して，後述する私書箱クラウドが Subscribe し，データを利用する各システムに分配する構成となっている．

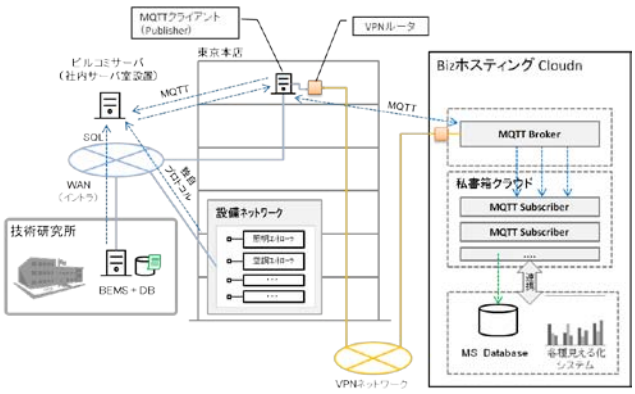

図 4　実証実験概要

3.1 システム概要

今回の検証では，セキュアなデータ送信のために VPN ネットワークを用いて検証用のビルとクラウドを接続した．図 5，表 1 に具体的な接続環境を示す．

3.1.1 私書箱クラウド

一般的にビルに適用される見える化システムなどのシステムは，MQTT 仕様に対応していないために，何らかのゲートウェイが必要となる．

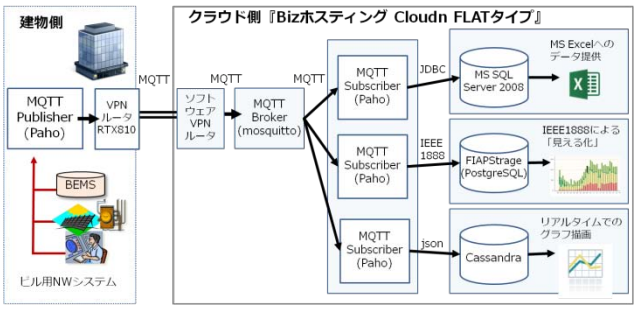

図 5　MQTT プラットフォーム

本検証では，Microsoft Excel でのデータ解析のために，MS SQL Server，IEEE1888 対応のために FIAP Storage，NoSQL による解析のために Apache Cassandra[14]に建物の計量データを保存したが，それぞれに対応したインタフェースを設ける必要があった．我々はこれらの管理するためのシステムとして，「私書箱クラウド」を構築し，各種システムや外部の MQTT Broker との接続や Topic の管理に利用することを考えた．

私書箱クラウドは Java で実装され，MQTT Broker に対して特定の Topic を Subscribe し各システムと連携するモジュールと，それらを格納するコンテナ，管理モジュールで構成される．なお，ビルコミのアーキテクチャにおいては，ren.Cloud にあたる．

表 1　実証実験環境

建物側		
ren.Gateway	OSGi : Equinox 1.2.0	
MQTT クライアント (Publisher)	名称 ： MQTT Publisher MQTT library : Paho ver. 0.4	
クライアント用マシンスペック	OS : Windows 7 Professional CPU : Intel® Core™ i5-3470 3.20GHz×2 RAM : 16.0 GB	
VPN ルータ	YAMAHA : RTX810	
回線	OCN 光「フレッツ」IP1（ファミリータイプ）	
クラウド側(右は Biz ホスティング Cloudn FLAT タイプのプラン名，OS)		
MQTT Broker	mosquitto 1.2.1	プラン: v2（2vCPU, RAM 4GB），vQ（0.25vCPU, RAM 0.5GB） OS : CentOS 6.4 64bit
MQTT クライアント (Subscriber)	名称 ： 私書箱クラウド MQTT library: Paho ver. 0.4	プラン: v2（2vCPU, RAM 4GB） OS : CentOS 6.5 64bit
見える化システム1	Microsoft SQL Server 2008 R2 Standard Edition	プラン: v1（1vCPU, RAM 2GB） OS : Windows Server 2008 R2 64bit
見える化システム2	FIAP Storage 2-20130509	プラン: v2（2vCPU, RAM 4GB） OS : Ubuntu Server 13.04 64bit
	実証実験用見える化システム	プラン: v2（2vCPU, RAM 4GB） OS : CentOS 6.4 64bit
見える化システム3	Apache Cassandra 1.2.13	プラン: v2（2vCPU, RAM 4GB） OS : CentOS 6.5 64bit
VPN ルータ	strongSwan 4.5.2	プラン: vQ（0.25vCPU, RAM 0.5GB） OS : Ubuntu Server 13.04 64bit

3.1.2 Topic の設計

本検証では，MQTT の Topic の命名規則として，表 1 に示す階層構造を採用した．どこで(where)，または誰が（who)を 5 階層，何を（what）を 5 階層で定義し，最後尾に読／書のパラメータを付与する．命名が困難な部分は，ハイフン「-」を用いる．また，発信するデータの内容はペイロード部分に記載される．ペイロードの内容は value（データの内容）, datetime（日時）を必須とし，そのほかに unit（単位）などをオプションとして定義している．
例えば，[takenaka.co.jp/Tokyo/THX/3F/SW1/HVAC/AHU/-/AHU3SW1/VAV2_heating_status/R] という Topic の場合に，[value=0&datetime=2014-01-20T15:37:21+0900] というペイロードが付与される．

表 2　MQTT Topic 命名規則

No	属性	Key	例
01	Where / Who	事業者 (ｻｰﾋﾞｽ提供者)	com / org / etc.
02		敷地 site (DC 名 data center)	THX / R90 / TAK-E /etc.
03		棟 building (ｻｰﾋﾞｽ名 service)	棟名
04		階 floor (機能 function)	階数 / outdoor
05		エリア area (ﾊﾞｰｼﾞｮﾝ version)	フロア分割貸事務所など
06	what	カテゴリ category	空調/照明/電力/防犯/防災/衛生
07		種別 kind	対象種別 温度/湿度/LED/
08		型 type	機器メーカ / 型式 など
09		名称 name	機器の識別名称 topic 内で一意
10		パラメータ名 key	温度 / 湿度 / 照度 など
11	R/W	読･書	R / W

4. 実証実験

我々は商用のクラウド環境における MQTT のスケーラビリティ，堅牢性，IEEE1888 との相互接続性を検証するために負荷および QoS の検証を行った．本節ではそれらの内容と結果を示す．

4.1　MQTT Broker の負荷検証

実際の計量データに加えて，建物側から大量のダミーメッセージを発生させ，MQTT Broker に送ることで負荷検証を行った．

具体的には，MQTT Broker はメッセージ送信速度を 10,000 件/分から 1,000,000 件/分までを目標に 3,000 秒間で順次に増やしながらメッセージを送信し，仮想サーバのリソースを変えて検証を行った（図 6，図 7）．

なお、本検証における MQTT の QoS モードは 0 である．QoS モードについては 4.2 節で述べる．

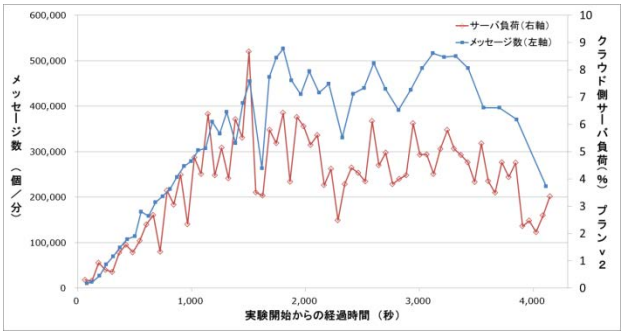

図 6　負荷検証試験（CPU 2 コア，メモリ 4GB）

図 6 は仮想サーバとしてプラン v2（CPU2 コア：2GHz 程度，メモリ 4GB）を用いた結果である．サーバ負荷，メッセージ数ともに比例関係で増加しているが，メッセージ数は 40 万件/分程度が上限で，平均として 30 万件/分程度で安定し，サーバ負荷（CPU の使用率）も 5%ほどで安定する．最大の負荷でも 9%程度であり，本システム構成においては充分に負荷を賄えるスペックであると考える．メッセージ数が飽和する理由としては，①ビル側の MQTT Publisher，②MQTT Broker，③仮想サーバ，④VPN ネットワークが考えられるが，本検証の結果だけでは判断できない．

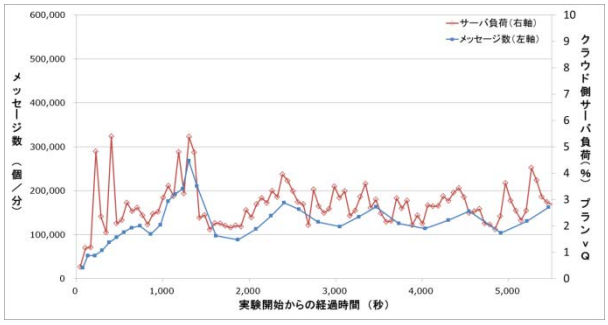

図 7　負荷検証試験（CPU 0.25 コア，メモリ 0.5GB）

図 7 は仮想サーバとしてプラン vQ (CPU0.25 コア：2GHz 程度を 4 台の仮想サーバで共有，メモリ 0.5GB）を用いた結果である．図 6 と同様の傾向を示しているが，メッセージ数は 20 万件/分ほどを上限として，平均 15 万件/分程度となり，サーバ負荷は 3%程度で安定する．

上記の 2 つの結果から，MQTT によるメッセージの最大送信量は仮想サーバの性能に依存していることが分かる．最大送信量に制限がかかる原因としては，上に述べたように原因が幾つか考えられるが，最も低廉なプランにおいても 20 万件/分程度までは送ることが可能であり，今回のような計量データだけであれば，充分な性能であるといえる．なお，建物設備の監視点数は現在 1/2~1/3 ポイント/㎡程度とおわれており，10 万㎡の大型ビルにおいても 4 万ポイントとなる．今後，ポイント数は増加すると言われているが，仮想サーバの増強などで対応可能である．

4.2 MQTT の QoS モードの性能検証

MQTT には QoS モードの設定があり，レベルとしては 3 段階 (0, 1, 2) 存在する．0 は 1 回限りのメッセージ送信で，TCP/IP で提供されるレベル以上の再送処理は行わない．1, 2 とレベルが上がるに従い，より確実に配信されるようになるが，プロセッサおよびネットワークのオーバーヘッドが増加する．

建物の計測データを扱う場合，それらは積算値として取得されることが多いために欠損は問題にならない場合が多いが，瞬時値や警報などの重要情報については，確実に配信されることが必要となる．そこで，QoS モードの違いにおけるメッセージ送信について検証を行った．なお，仮想サーバの環境としては，それぞれ図 6 と同様のプラン v2 を利用した．

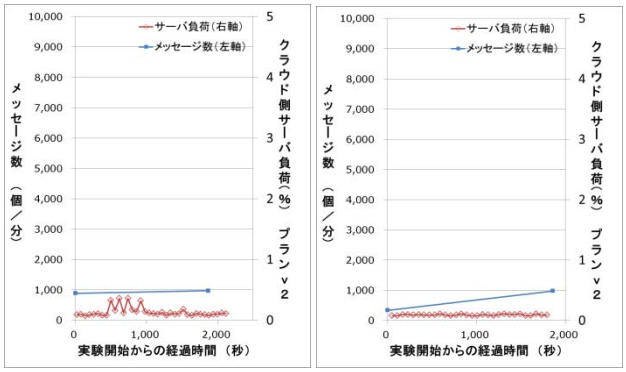

図 8　QoS モード検証試験

結果としては，図 8 に示す通り，QoS=1, 2 のそれぞれにおいて，サーバ負荷は 0.5%以下であるものの，最大メッセージ数は 300～1000 件/分程度となった．QoS に実装に依存した結果である可能性があるが，リアルタイムの設備制御やエネルギーの見える化を行う目的において，性能として不充分と考える．警報などの重要情報のみを対象とするべきだろう．

なお，MQTT Publisher と私書箱クラウドのそれぞれにロガーを設置し，ログ照会を行うことで，QoS=0 におけるメッセージの欠損割合の検証も行ったが，欠損はほとんど認められなかった．私書箱クラウドが何らかの理由で停止した際も，MQTT Broker の Retain 機能を使うことで，直近のメッセージは保持されるため，一般的な計量データの送信には QoS=0 を用いるべきと考える．

4.3 IEEE1888 検証

我々は既開発の見える化システム（図 9）を IEEE1888 対応とし，図 10 の構成で MQTT との連携を試みた．

事前の検証として，システムへの計量データ取り込みの時間計測を行い，システム最適化の切り口を探った（図 11）．環境データ（計量データ）の受信にかかる処理が全体の処理の 80%程度となることが分かり，これらの最適化が課題といえる．そのため下記の検証を行った．

1） 1 回の FETCH で照会するポイント数についての検証
2）集計するポイント数の増加についての検証

結果として，FETCH するポイント数を調整することでサーバ負荷を低減させることができること，MQTT によるリアルタイムデータとも問題なく連携できることが検証された．

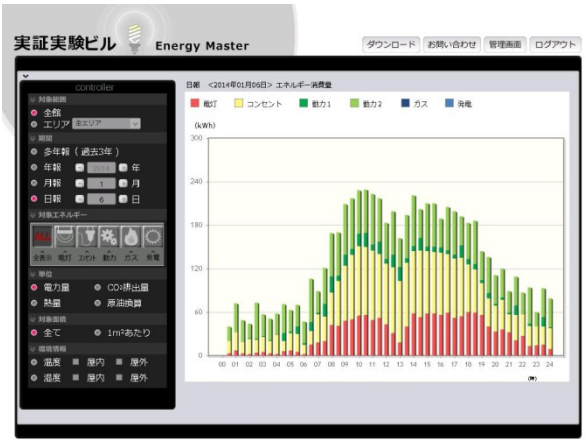

図 9　見える化システム

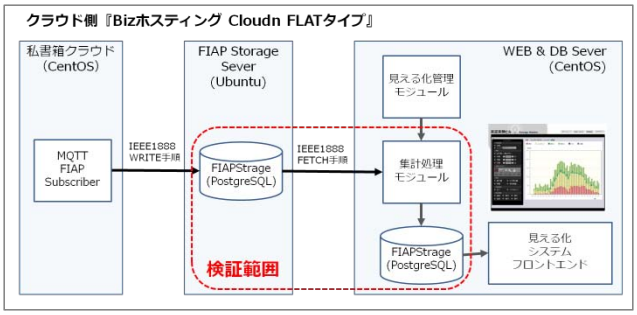

図 10　見える化システムの構成

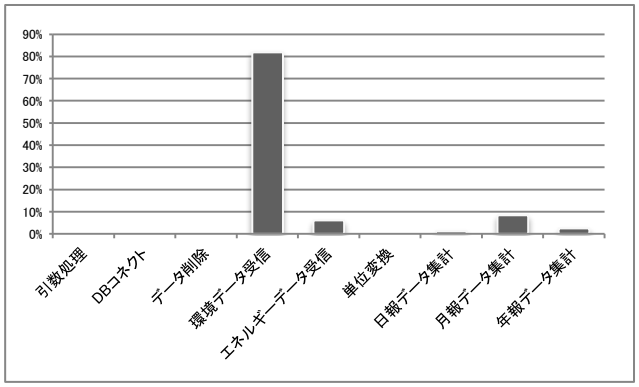

図 11　見える化の処理時間割合

なお，今回は検証においては，私書箱クラウドにおける MQTT から IEEE1888 へ変換は 1 メッセージあたり 1 つの対応で全て行った．IEEE1888 は HTTP+XML であるためオーバーヘッドが大きく，それらの変換の負荷検証も行うべきではあるが，4,000 ポイント/分程度のデータ量であれば，安定して動作していたために，本検証では FIAP Storage から IEEE1888 で FETCH する件数，ポイント数を中心に検証を行った．

4.3.1 FETCH で照会するポイント数についての検証

図 12 は集計するポイント数（環境データ数）を固定し（441 ポイント），1 回あたりのポイント照会件数を変動させた場合の処理時間とメモリ使用量を示している．

結果として，概ね 50～100 件ずつ処理するケースが効率的であることが分かる．なお，本検証で用いた見える化システムは 30 分間の集計データを扱うために，IEEE1888 では 30 分毎のデータを抜き出すように，周期的にクエリ（IEEE1888 FETCH 手順）を生成している．

4.3.2 集計するポイント数の増加についての検証

図 13 は集計するポイント数を仮想的に増加させた際の処理時間を示している．横軸は図 12 と同様で，一回の FETCH で照会するポイント数を表す．集計ポイントの増加に従い，処理時間が増加する傾向がみられるが，照会ポイントの調整により，処理時間を短縮できることが分かる．なお，メモリ使用量についても同様の傾向が見て取れる．

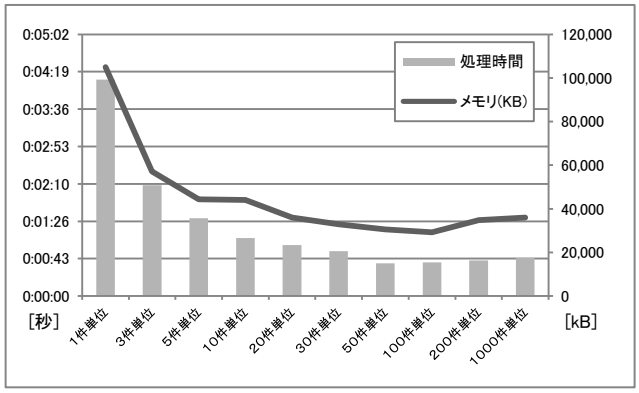

図 12　ポイント照会件数による性能評価試験

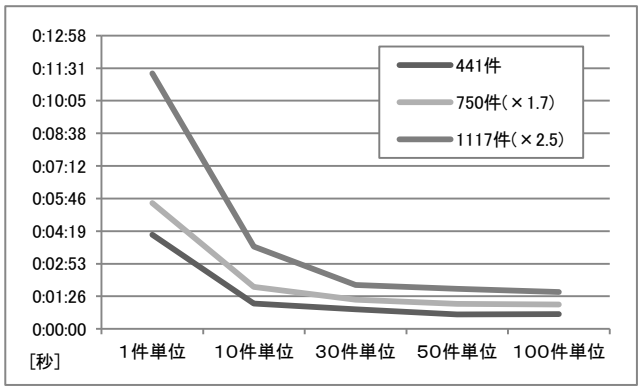

図 13　集計するポイント数による性能評価

5. 評価

我々はビルコミに対する与件として，リアルタイム性，スケーラビリティ，堅牢性，ユーザビリティ，相互接続性を設定した．それらの評価を表 3 に示す．検証結果により，それぞれの項目において充分な性能を持つといえる．また，MQTT のプラットフォームは建物単体だけではなく，様々な情報が大量に飛び交うことが予想されるスマートシティのインフラとしても充分な機能と性能を持つといえるだろう．

表 3　建物設備における MQTT プラットフォームの評価

リアルタイム性	MQTT の特徴として，軽量なメッセージングプロトコルということがある．本検証以外でも様々なデモシステムの製作，展示を行い，同様のクラウド環境下においてもリアルタイムにシステムが稼働することが確認できた．
スケーラビリティ	本検証において，低廉なクラウドのプランにおいてメッセージ送受信で 20 万件／分の性能が出ることが確認できた．処理件数は仮想サーバの性能に依存するため，多棟管理となった場合，M2M と統合した場合にも充分なスケーラビリティを持つと考える．なお，MQTT Broker にはロードバランサやクラスタなど，多くの負荷分散機能が定義されている．それらを組み合わせることでより多くの情報を扱うことができると考える．
堅牢性	MQTT の QoS モード検証において，QoS=1,2 はリアルタイムの情報提供に不向きなことが検証されたが，QoS=0 でもほとんど欠損無く送受信が行われることが分かった．建物に関わる情報の多くを占める計量データは積算値であるため，粒度の細かい情報において欠損は大きな問題にならないともいえる．充分な堅牢性と判断できる．
ユーザビリティ	ビルコミは OSGi を利用しているために，遠隔からのアプリケーションの導入，管理が簡便に行える．ユーザビリティについては，アプリケーションの実装によるが，ビルコミではユーザの操作をリアルタイムに建物設備にフィードバックすることが可能である．
相互接続性	ビルコミや私書箱クラウドを使うことで，今回検証を行った IEEE1888 の他，BACnet，JDBC，NoSQL エンジンなどとの連携も簡便に行うことができた．サーバ負荷については，仮想サーバのスペックを調整することで柔軟に対応することができる．

6. 関連研究

建物内外の様々な情報を用いて，設備制御などを行うプラットフォームの研究としては，U.C.Barkeley の SDB（Software Defined Buildings）プロジェクトがある[15]．BOSS[16](Building Operation System Services)と呼ばれる建物設備制御のためのフレームワークの開発や，建物内のセンサなどのメタ情報と時系列データを効率的に扱うことができる sMAP[17]（the Simple Measurement and Actuation Profile）などの開発と実証を行っている．BOSS は建物設備の OS として機能し，専用のスクリプト等を用いて建物設備の制御を行うなど，施設の管理者にも簡便に利用できるように設計されている．また，我々は実空間をモデリングした 3D 空間から直接照明などの機器を制御したり，スクリプトによって自動制御したりする取り組みも行っている[3]．

ビルコミは MQTT と OSGi を利用しているために，上述のシステムと比べて，リアルタイム性や拡張性に優れているといえるが，スクリプト等への対応については今後の課題と考えている．

なお，海外などでは M2M のための商用サービスとして MQTT を用いた M2M 基盤の提供が始まっている[18]．日本にも進出を始めているが，課金の考え方が従量課金である など，本検証のように常時大量のトラフィックを扱うシステムには不向きと考える．

7. まとめと今後の展開

本検証では MQTT プラットフォームを，建物設備およびクラウド・VPN 環境に適用する試みを行い，ビルコミや私書箱クラウドを用いて IEEE1888 を利用した見える化システムとの連携を行った．その中でスケーラビリティ，堅牢性に関する検証を行い，MQTT が建物単体だけでなく，複数ビル更にはスマートシティへの展開にも充分な性能を持つことを検証した．ビルコミや私書箱クラウドを利用することで，MQTT の特徴であるリアルタイム性に加えユーザビリティや相互接続性についても確保することができる．

今後はこれらのプラットフォームを用いて，ビッグデータ解析などを組み合わせ，更なる付加価値を産むための技術開発やサービス開発・提供を行う方針である．また，今回は VPN を利用したセキュアな環境で検証を行ったが，スマートシティなどを想定すると，モバイル環境やインターネット上に分散した様々なリソースのデータともセキュアに連携させる必要性がある．そのためのアーキテクチャやセキュリティの検証も今後の課題である．

参考文献

[1] MQTT (MQ Telemetry Transport) , http://mqtt.org/
[2] 吉田薫, 江崎浩, “グリーン東大工学部プロジェクトにおける取り組みと成果”, 電子情報通信学会電子情報通信学会信学技報 Vol.109, No.351 (2009)
[3] 松岡康友,”メタバースを用いた建築制御フレームワークの可能性 －遠隔操作の応答時間検証－”, 日本建築学会　第 31 回情報・システム・利用・技術シンポジウム論文集 (2008)
[4] OSGi , OSGi Alliance, http://www.osgi.org
[5] BACnet (Building Automation and Control Networks) , http://www.bacnet.org/
[6] Lonworks, LONMARK JAPAN , http://lmjapan.org/
[7] OpenADR, OpenADR Alliance, http://www.openadr.org/
[8] 金子弘幸, “オフィス内のワーカーの活動量比較　レーザーセンサーを用いた行動モニタリング調査　その 1 ”, 日本建築学会学術講演梗概集 (2012)
[9] Biz ホスティング Cloudn, http://www.ntt.com/cloudn/
[10] strongSwan, http://www.strongswan.org/
[11] JMS (Java Messaging Service), http://stomp.github.io/
[12] STOMP, http://stomp.github.io/
[13] Kura Project, http://www.eclipse.org/proposals/technology.kura/
[14] The Apache Cassandra Project, http://cassandra.apache.org/
[15] Software Defined Building, http://sdb.cs.berkeley.edu/sdb/
[16] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David Culler, “BOSS: Building Operating System Services”, Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (2013)
[17] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler, “sMAP - a Simple Measurement and Actuation Profile for Physical Information.”, Proceedings of the Eighth ACM Conference on Embedded Networked Sensor Systems (2010)
[18] Everyware Device Cloud, Eurotech, http://www.eurotech.com/en/products/software+services/everyware+device+cloud/

† 株式会社竹中工務店　Takenaka Corporation
‡ エヌ・ティ・ティ・コミュニケーションズ株式会社
NTT Communications Corporation