修士論文　　2012 年度 (平成 24 年)

# 応答集約機能を有する CoAP proxy による センサーネットワーク輻輳の軽減

慶應義塾大学 政策・メディア研究科

米村 茂

修士論文要旨

2012 年度 (平成 24 年度)

# 応答集約機能を有する CoAP proxy によるセンサーネットワーク輻輳の軽減

実空間をコンピュータシステムで扱う，いわゆるモノのインターネット (Internet of Things) では，様々なモノが様々なアクセス方式でネットワークに接続される．アクセス方式が異なるモノに対する情報サービスの提供方法としては Web 技術の応用が有望である．しかし，比較的低速なネットワークを用いる環境 (Constrained network) では，HTTP を用いた既存のプロトコルはオーバーヘッドが大きく不向きである．これらの制約環境下での利用に特化した Web プロトコルとして，Constrained Application Protocol (CoAP) が IETF において標準化作業進行中である．CoAP を用いることにより，モノとの End-to-End の通信がアクセスネットワークに拠らず可能になる．しかし，一方で Constrained network 区間では，リクエストの増加に伴い輻輳が発生し，一つのモノが関係できるサービス数が制限される．本論文では，HTTP および CoAP を用いてデータを収集する環境を構築し，情報システムにおける性能ボトルネックを実験的に調査し，定期応答メッセージが輻輳の原因となることを示す．その解決案として，ゲートウェイ装置がアプリケーションデータの内容を解釈すること無しに，センサーノードの応答集約を可能とする手法を提案する．提案手法は ZigBee を用いた実装により，同じリソース情報を 1 秒間隔で送信する応答について，1 つのセンサーデバイスが応答数が 8 の場合において，通信量を 6464 bps から 976 bps へと低減でき，4 つのセンサーデバイスが連携できるクライアントの数を合計 31 から 60 へと改善できることを明らかにした．

キーワード

1. ワイヤレスセンサーネットワーク, 2. CoAP, 3. ネットワークの輻輳, 4. 集約手法

慶應義塾大学 政策・メディア研究科

**米村 茂**

Abstract of Master's Thesis

Academic Year 2012

# Mitigation of network congestion with response aggregation function in CoAP proxy

Various things and services are connecting each other with various network types. This is a typical communications model of Internet of Things. The use of Web technology as the foundation of IoT is promising because the Web technology works on top of different OS on different networks. But, existing Web protocols, such as HTTP, do not work well in a constrained network such as IEEE802.15.4. A working group in IETF is standardizing Constrained Application Protocol (CoAP) as a generic Web protocol in a constrained environment. In this thesis, I build an End-to-End communications system with HTTP/IP and CoAP/ZigBee networks which composed on over 60 sensor devices spread in several geographical locations. At the every intersection of IP and ZigBee we need to have a proxy to exchange the HTTP/CoAP layer. The use of CoAP facilitates the development of the proxy because the proxy transparently transfers application data by simply exchanging HTTP and CoAP. In our measurement, the maximum number of application clients is 31 when the number of devices on the constrained network is 4 and the time interval of data notification is 1 second. This is insufficient to general home applications. To solve the problem I propose a response aggregation method without losing the End-to-End nature. This is done by two layer numbering for a subscription. Since the data notification are aggregated, the method can mitigate congestion in the constrained network. In our implementation the reduction of the traffic volume was from 6464 bps to 976 bps with an one device, and the function can increases the number of client applications related with 4 devices from 31 to 60.

Keywords :
1. Wireless Sensor Network, 2. CoAP, 3. Network congestion, 4. Aggregate function

Keio University, Graduate school of Media and Governance
Shigeru Yonemura

# 目次

# 図目次

# 表目次

# 第 1 章

# 序論

## 1.1 研究の背景

今日，ICT（Information and Communication Technology）は様々な技術革新を経て成熟し，我々の日常生活に深く浸透し社会インフラとしての重要な位置にある．ICT システムの中でも実空間をコンピュータシステムで扱う，いわゆるモノのインターネット (Internet of Things : IoT) では，様々なモノが様々なアクセス方式でネットワークに接続される．一つのモノであっても製造段階やサプライチェーンにおいてはバーコードや RFID で情報システムに認識され，家庭などでユーザーが使用する時には WiFi や Ethernet を通じてシステムに接続するといったように，製品のライフサイクルの過程で利用するアクセス方式が変わることも考えられる．このように様々なアクセス方式のモノを扱う情報サービスの提供方法としては，Web 技術の応用が有望である．エネルギーマネージメントやビルディングオートメーション，スマートハウス，その他の Machine-to-Machine などのアプリケーションについても，今後 Web を通じた通信が可能となることが望まれている．また，これらのアプリケーションは家庭では照明，室温などの環境情報や，人の心拍，血圧，体温を，人の進入しづらい災害地では温度，煙などの多様なセンサーデータを低コスト，低消費電力で遠隔にて収集，モニタリングを可能とすることが求められるため，ワイヤレスセンサーネットワークの普及が必要不可欠である．

ワイヤレスセンサーネットワークのようにデバイスの通信性能が低い場合や，ZigBee など比較的低速なネットワークを用いる環境 (Constrained network) においては，HTTP を用いた既存のプロトコルはオーバーヘッドが大きく不向きである．これらの制約環境下での利用に特化した Web プロトコルとして，Constrained Application Protocol (CoAP) が IETF において標準化作業進行中である [1]．CoAP を用いることにより，

CoAP により既存の Web サービスと同様にモノとの End-to-End 通信がアクセスネットワークに拠らず可能となる．End-to-End 通信はアクセスネットワークを変換するゲートウェイ装置の単純化や，モノと関連するサービスの幅を広げることができるメリットがある．しかし，一方でモノの関連サービスが増加することで，Constrained network 区間ではセンサーデバイスへのリクエスト数が増加し，ワイヤレスセンサーネットワーク区間での通信量が増大するのに伴い輻輳が発生し，システムの健全性が失われる．Internet of Things システムの寿命を長く保つためには，この問題の解決が求められる．

## 1.2 本研究の目的

本研究では、まずクライアントがセンサーノードから HTTP および CoAP を用いてデータを収集する環境を構築すると共に，情報システムにおける性能ボトルネックを実験的に調査し，その結果からセンサーノードの定期応答メッセージがセンサーネットワークにおける輻輳の原因となることを示す．そして，この問題に対する解決案として，ゲートウェイ装置がアプリケーションデータの内容を解釈すること無しに，センサーノードの応答集約を可能とする手法を提案し，実装によってその有効性を明らかにする．

## 1.3 本論文の構成

本論文の構成は全 8 章からなる．第 2 章では,実空間コンピュータにおける End-to-End 通信の有効性についていくつかの例と共に述べる．第 3 章では，ワイヤレスセンサーネットワークと，そのような Constrained Network を含むシステムでの利用に特化した Web プロトコルである CoAP について述べる．第 4 章では，ワイヤレスセンサーネットワークにおける通信についてシミュレーションによって考察し，HTTP および CoAP を用いたデータ収集環境における実測実験によって，センサーネットワークにおける輻輳について考察する．第 5 章では，ワイヤレスセンサーネットワークにおいて輻輳を軽減する手法の一つである，応答集約についての関連研究を示す．第 6 章では，輻輳を解決するための提案手法について述べる．第 7 章では，提案手法の有効性評価について述べる．第 8 章では本研究の結論について述べる．

# 第 2 章

# End-to-End 通信

本章では，Internet of Things システムにおける End-to-End 通信の重要性について述べる．また，End-to-End 通信を行うシステム設計の例として Smart Energy Profile 2.0 と RACOW を示す．

## 2.1 End-to-End 通信の必要性

Internet of Things システムに参加するデバイスとそのアクセス方式は多様である．例えば，家庭内の家電について考える．テレビやビデオデッキは Ethernet で接続できるものが存在し，体重計や血圧計などについては，WiFi や Bluetooth で接続するものも存在する．一方で，照明機器などの有線での接続が困難なものや，古い家電のようにネットワークへのアクセス手段を持たない家電も存在する．家庭内に存在するのこれらの家電全てを接続しようと考えた場合には，通信機能のない家電には後付けでネットワーク機能をつけるなどの対処が必要となる．例えば，電源プラグに付けるスマートメーターなどである．また，家庭全体にネットワークを巡らすことや電源に繋がない機器の参加を考えると，マルチホップネットワーク，低消費電力を特徴とする ZigBee などのワイヤレスネットワークの利用が有望である．このように家庭という限られた空間においても，そこに存在するモノのネットワークアクセス方式は多様である．

IoT システムではこれらの多様なアクセス方式にまたがって，センサーデータの収集，モニタリングや操作などの相互通信を行う．つまり，IoT システムはそれぞれのデバイス，アクセス方式の特徴に合わせて設計される必要がある．この要求を満たす方法の一つとして，アクセスネットワーク間で変換処理を行うゲートウェイ装置において，それぞれのアクセス方式に適するように変換処理を行う手法がある．この手法はデバイス，アク

セス方式，サービスのそれぞれ特徴に最適化した動作を可能とする．例えば，IP ネットワークと ZigBee ネットワーク間のゲートウェイ装置において，アプリケーションデータを解釈し，ZigBee 区間においてはバイナリデータとして扱うという手法を取ることができる．しかし，モノが複数のサービスと関係する環境においては，全てのゲートウェイ装置においてアプリケーションごとの機能を持つ必要があり，ゲートウェイの機能が複雑化する．また，多くのアプリケーションが展開できる IoT システムを目指す場合には，アプリケーションの設計者がゲートウェイ装置での処理を考えなくてはならないことは，新規アプリケーション追加の障害と成る．

本研究では多様なアクセス方式を含む IoT システムにおいてゲートウェイ装置のコストを下げる手法として，アプリケーションとセンサーノードの End-to-End 通信に着目する．End-to-End 通信では，ゲートウェイ装置はアプリケーションデータの解釈を行わず，アクセス方式の変換のみを行う．アプリケーションデータの解釈はシステムの両端であるアプリケーションとセンサーノードによってのみ行う．そのため，サービス提供者はアプリケーションの設計時にゲートウェイ装置を考慮せずに済む．

以下に End-to-End 通信を実現するための設計をいくつか示す．

## 2.2 Smart-Energy Profile 2.0

Smart-Energy Profile 2.0 (SEP 2.0) はスマートグリッドの実現を目的として，家庭内のスマートエナジーデバイスを相互接続するプロトコルと標準を作成するために，考えられた策定活動中の規格である [2]．元は ZigBee Alliance と HomePlug Alliance が TCP / IP とそれに準ずる MAC 層，物理層の上で，それぞれの規格動作させるための規格として生まれた．現在もこの思想は反映されており，下層のプロトコルはアプリケーションと密接に関わる場合を除き基本的に指定せず，802.15.4 や 802.11, 1901 など様々な物理層の上で動作する．現在は WiFi Alliance や HomeGrid Alliance も加わり，様々な電気機器のアプリケーション層における相互接続性を目指している．

SEP 2.0 プロトコルはスマートエナジーシステムを実現するために必要となる，デバイスの必要最低限の機能要件とその表現方法に関して定義している．基本的な機能としては計測や価格決定，需要応答による負荷調整などがある．これらの機能は消費者に情報を提供することで家庭内の消費電力の管理を可能とし，スマートグリッドにおいてはピーク時の消費電力調整を可能とする．さらに，電気自動車やエネルギーの再分配などの，より発展的なユースケースに付いての機能要件についても定義している．

計測機能はデバイスがスマートメーターやその他の計測機能を持つデバイスに向けた機

能である．SEP 2.0 のアプリケーションはスマートフォンやタブレット，専用の表示デバイスにおいて動作することを前提としており，家庭全体のリアルタイムの電力消費情報を利用者に提示できる．この情報はスマートメーターから直接取得することも，クラウドベースのサーバーから間接的に取得することもできるし，計測機能を持つ家電から直接取得することも可能である．

SEP 2.0 アプリケーションプロトコルは広く普及している HTTP Web サービスと同じように，REST 構造に基づいて設計されている．REST 構造はクライアントーサーバーモデルを基本としており，サーバーがリソースの管理を行う．クライアントはサーバーに read, write, create, delete のようなコマンドを用いて，サーバーのリソースに対してリクエストを出す．SEP 2.0 におけるリソースはメーターの値や，電気料金，需要応答を扱うイベントなどに該当する．

## 2.3 RACOW プロジェクト

RACOW とは，RFID Auto-Commissioning Open system with WiMAX の略称であり，このプロジェクトは総務省平成 21 年度第 2 次補正予算「ネットワーク統合制御システム標準化等推進事業」における WiMAX を利用したデータ収集システムによる環境負荷低減の実証を行うプロジェクトである [3]．RACOW プロジェクトでは家電が工場で生産され，流通によって搬送され，小売店により販売され，家庭において使用され，中古市場により再利用され，廃棄されるまでの家電のライフサイクルとそれに付随する情報の動きを管理する，家電のライフサイクルマネージメントの基盤提唱がされている．家電製品は製造時から，その識別のためにユニークな ID が用いられており，この ID とその家電が作られれた状況や搬送，使用状態などの情報をひもづけて収集する．RACOW プロジェクトではこれらの情報を利用して，各家電の利用情報共有や故障対応時の事前情報取得などのアプリケーションが実現でき，環境負荷低減の効果があることが示されている．

### 2.3.1 RACOW システム

Auto-ID Lab Japan では RACOW システムの実証実験のために，50 程度の家電とセンサーデバイスを 4 カ所の地理的に離れた場所に設置し，大学のエネルギー監視や家電の遠隔操作システムとして運用している．それぞれの場所は，Ethernet と ZigBee を利用している．各家電またはセンサーデバイスはセンサーモジュールを持っており，管理しているセンサーの数は 180 以上である．各サブネットには図 2.1 に示すように，IPv6 を用

いて Ethernet または WiMAX で接続できる．

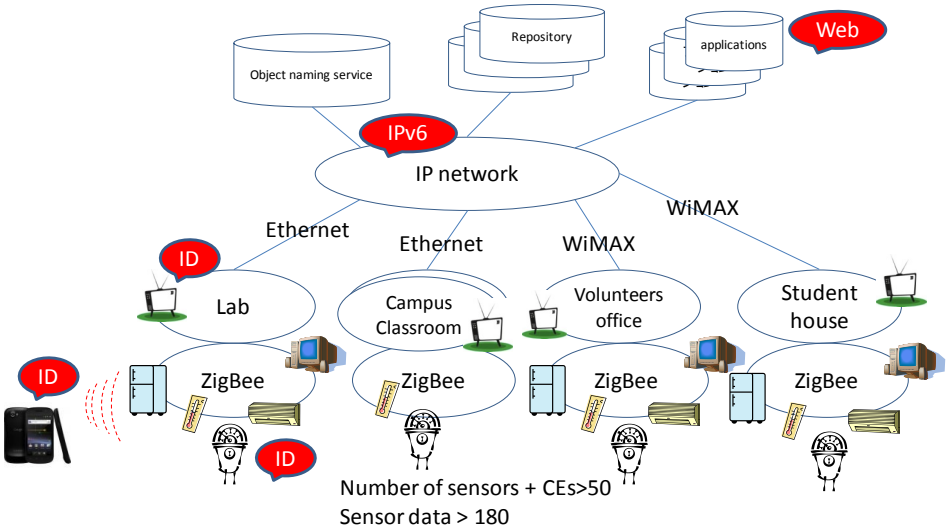


図 2.1: RACOW 実証実験環境

この実験ネットワークはホームネットワークの機能を含んでいる．コントロールポイントや UPnP デバイス，ゲートウェイ，ZigBee Coordinator，エンドデバイスなどがこれにあたる．図 2.2 に示すこのネットワークを構成する機器について述べる．

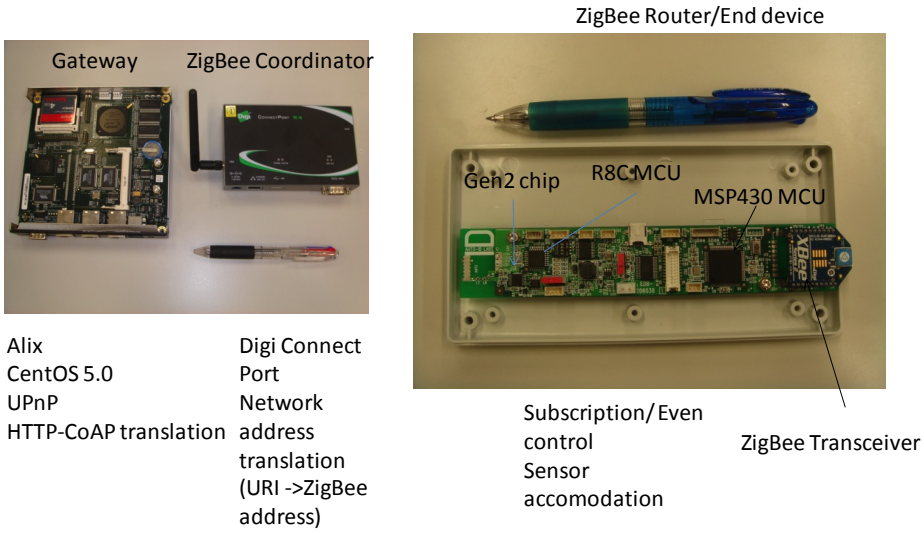


図 2.2: RACOW 機器

図 2.2 の右のセンサーボードはシステム ZigBee ルーター，エンドデバイスとして用いたテストボードである．このボードには R8 MCU，MSP430 MCU，ZigBee トランシーバー，Gen2 チップが搭載されている．このボードの特徴的な点は図 2.3 に示すデュアルインターフェイス RF タグを内蔵しているところである．

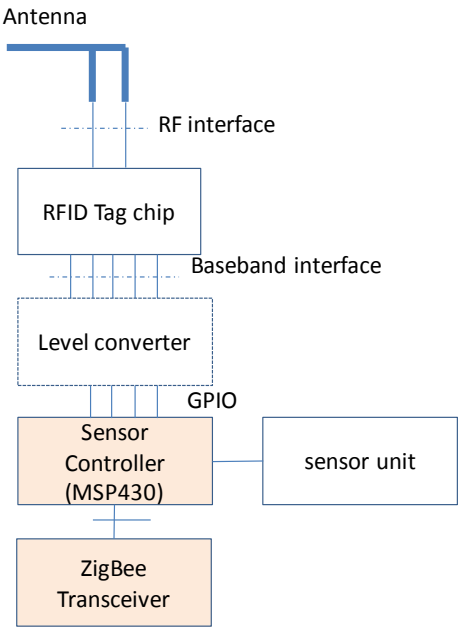


図 2.3: デュアルインターフェース RF タグ

デュアルインターフェイス RF タグは，RFID インターフェースとベースバンドインターフェースを持っており，どちらからでもタグチップ内のデータの読み書きが可能である．このボードでは，ベースバンドインターフェースをセンサーコントローラーと接続している．つまり，ZigBee ネットワークを IP ネットワークと接続することにより，RFID タグチップ内の情報を ZigBee そして IP を通して扱える．生産過程や流通過程，販売過程においては，RFID インターフェースを用いて通信を行い，家庭に設置された後には UPnP を用いたオートコミッション機能により，アプリケーションとの通信が可能となる．

ZigBee コーディネーターはシリアル通信でゲートウェイと通信し，プロトコルの変換のみを行う．

ゲートウェイはボードコンピュータの PC エンジン Alix であり，このネットワークで言うところの ZigBee に当たる，非 IP ネットワークに所属するデバイスの IP ネットワークへの仮想デバイスを提供する．つまり，ゲートウェイは IP 通信のエンドポイント

である．この仮想デバイス機能により，アプリケーションは各家電との通信の足回りについて意識することなく，各デバイスが持つ通信モジュールが Ethernet や WiFi のような IP 機器であっても，ZigBee や Bluetooth のような非 IP 機器であっても同様に通信が可能となる．

また，ZigBee デバイスとの通信はアプリケーション層において End-to-End 通信であり，この通信の変換は図 2.4 に示すように行われる．図の通り，アプリケーションと端末である家電との間ではデータ XML がやりとりされ，通信経路上のゲートウェイやコーディネーターにおいては，アプリケーションデータに関する処理は行われない．

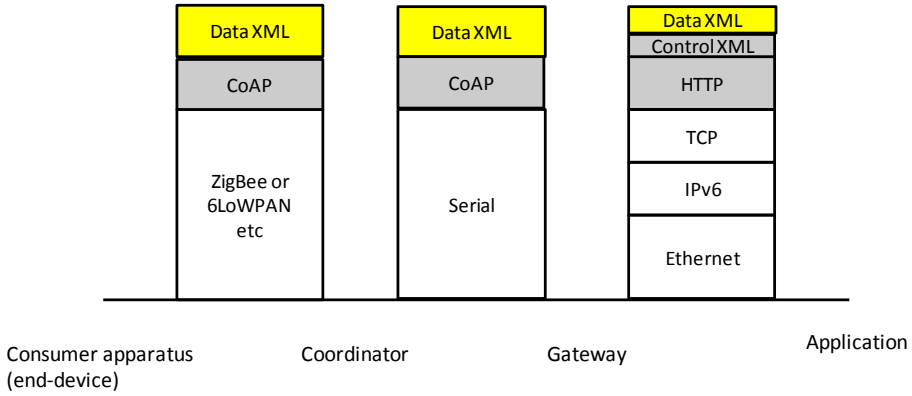


図 2.4: RACOW システムのプロトコルスタック

### 2.3.2 RACOW システムの定期応答通信

RACOW システムでは，アプリケーションがセンサーデバイスの状態変化を定期的に取得するためにサブスクライブコマンドが用意されている．サブスクライブコマンドを用いた定期応答通信は図 2.5 に示すように行われる．アプリケーションは定期的にセンサーデバイスのセンサーデータを取得したい場合，HTTP クライアントとして目的のセンサーデバイスを配下に持つゲートウェイ装置に対して，サブスクライブリクエストを POST する．このサブスクライブリクエストは，URL として図に示すようにゲートウェイ装置のアドレス (gw-uri) とセンサーデバイスの識別番号 (epc) を含む．また，サブスクライブセッションを識別するための Subscribe ID (sID)，応答の送信先である reply-uri(r-uri) を含む．POST を受けたゲートウェイ装置はアプリケーションに対して HTTP RESP を返信した後に，センサーデバイスに対してリクエストメッセージを送信する．このメッセージには，ゲートウェイ装置が送信したメッセージを判別するための Event ID (eID)

と，センサーデバイスの epc を図に示すように含む．ゲートウェイ装置は sID と eID の対応関係を保持することで，アプリケーションとセンサーデバイスのセッション関係を記憶する．リクエストメッセージを受信したセンサーデバイスは内部のリストに eID を追加する．センサーデバイスは一定時間が経過すると eID リストを元に応答メッセージを送信する．この応答メッセージには eID とセンサーデータが含まれる．応答メッセージを受信したゲートウェイ装置は sID と eID の対応関係リストを元に，HTTP POST メッセージを作成し，HTTP クライアントとして送信する．この POST メッセージの URL は r-dst と epc を含む．また，メッセージには sID が含まれる．

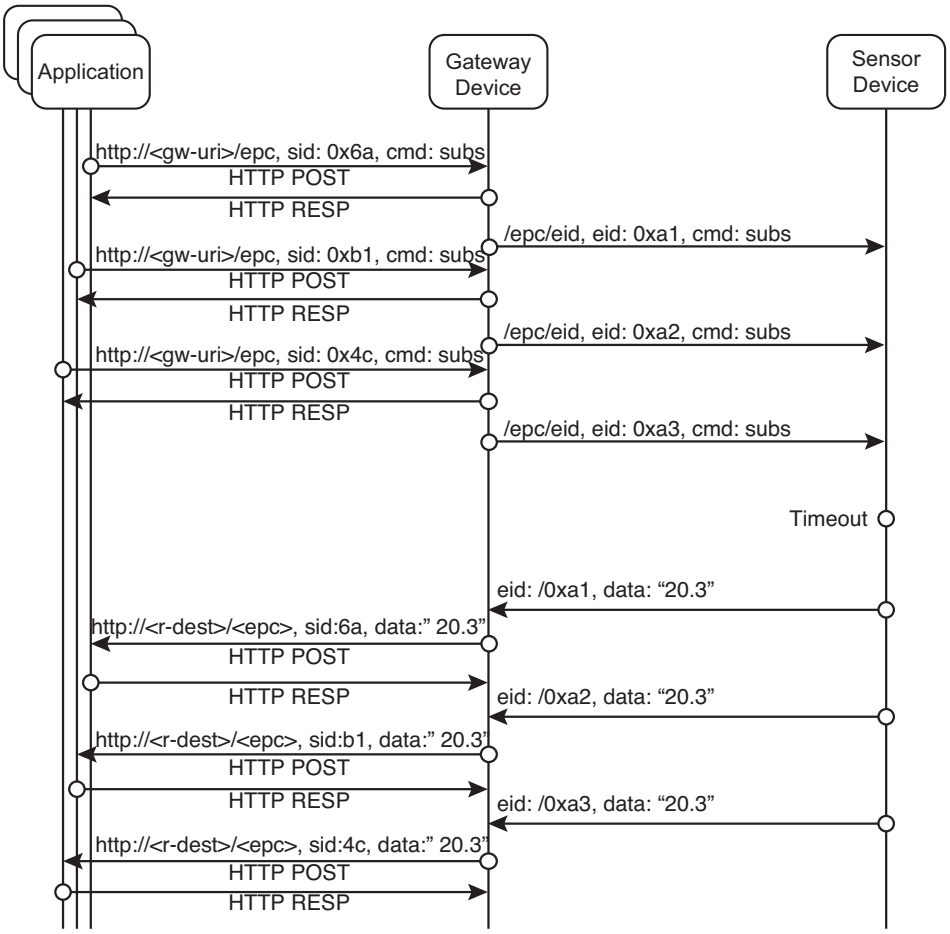


**図** 2.5: RACOW **システムのサブスクライブシーケンス**

## 2.4 本章のまとめ

**本章では，様々なアクセス方式のモノを含む** IoT **システムにおいて，**End-to-End **通信がサービス提供者の新規アプリケーション追加を簡易にすることを述べ，また，その** End-to-End **通信を用いたいくつかの設計を示した．次章では，**IoT **システムにおいて重要な位置にあるワイヤレスセンサーネットワーク技術と，**End-to-End **通信を異なるアクセス方式にまたがって行うための技術について述べる．**

# 第 3 章

# 関連技術

本章では，IoT システムの基盤技術であるワイヤレスセンサーネットワーク技術である，IEEE 802.15.4，ZigBee について述べ，また，複数のアクセス方式にまたがって End-to-End 通信を行うための技術である CoAP について述べる．

## 3.1 IEEE 802.15.4

IEEE 802.15.4 は低消費電力なワイヤレス通信のプロトコルであり，IEEE により仕様化され，標準規格となっている．IEEE 802.15 シリーズはワイヤレスデータ通信のプロトコルを定めたプロトコルシリーズであるが，IEEE 802.15.4 はその中でも特に組み込み機器などの相互通信のために設計されている．その目的のために，低データレートで省電力の物理通信方式を利用し，複雑でないプロトコル設計となっている．IEEE 802.15.4 では構成される近距離通信のネットワークは WPAN (Wireless Personal Area Network)，または PAN (Personal Area Network) と呼ばれる．IEEE 802.15.4 は Physical layer (PHY 層) と Medium Access Control layer (MAC 層) について定めている．

PHY 層

IEEE 802.15.4 の PHY (物理) 層について述べる．IEEE 802.15.4 は ISM (Industry-Science-Medical) 目的の周波数帯である，2.4 GHz 帯と 900 MHz 帯が使用でき，日本では 2.4 GHz 帯のみが使用可能である．機器が技術基準適合証明を受けていれば無線局免許を受ける必要なく利用できる．実測によると，屋外で 100m 程度，屋内では数十 m 程度の距離で通信が可能である．

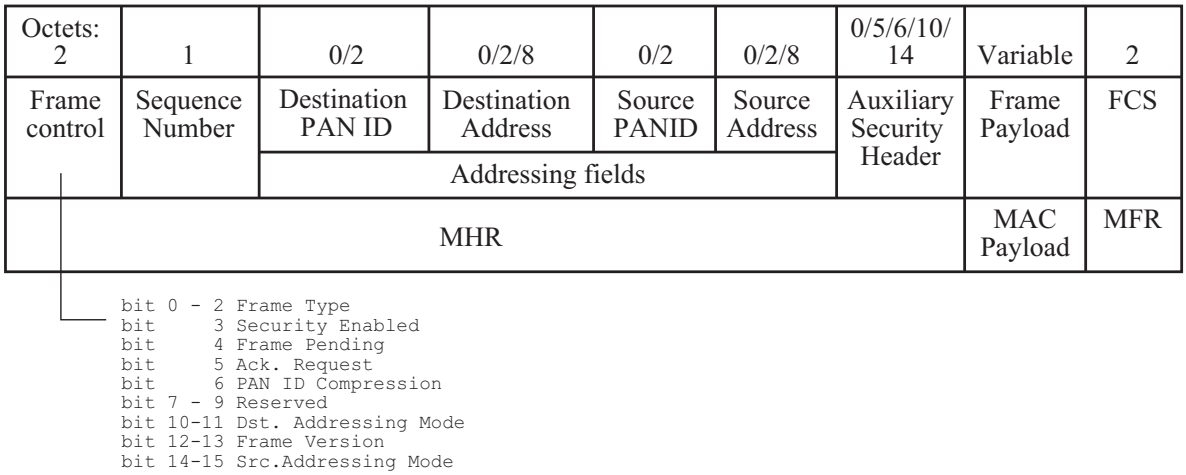


**図** 3.1: IEEE 802.15.4 MAC **フレームフォーマット**

MAC **層**

IEEE 802.15.4 **の** MAC (Medium Access Control) **層について述べる．まず，** IEEE 802.15.4 **で使用する** 2 **種類の識別子を述べる．**

- PAN ID IEEE 802.15.4 **のネットワークである** PAN **の識別子．**16 **ビット長．**
- MAC **アドレス** PAN **内でのノードの識別子．**PAN **内で一意な** 16 bit **長のショートアドレスと，**IEEE **から割り当てを受ける** 64 bit **長のロングアドレスの** 2 **種類がある．**

MAC **フレームのフォーマットを図** 3.1 **に示す．**PAN ID **及び** MAC **アドレスのそれぞれについて，** Source **および** Destination **が指定可能である．ブロードキャスト時には** PAN ID**，**MAC **アドレス (ショートアドレス) 共に宛先アドレスとして** 0xffff **を利用する．必要がない場合は省略も可能であり，またロングアドレス，ショートアドレスの選択はヘッダーの中の** Frame Control **にて指定する．**

### 3.1.1 ZigBee

ZigBee **プロトコルスタックは他のプロトコルスイートと同様に，いくつかの層で構成される．**PHY **と** MAC **は** IEEE 802.15.4 **のものを利用する．その上位に，マルチホップでネットワークを構成するための** Network (NWK) **層がある．そのさらに上位の** Application (APL) **層は** Application Support Sub-layer (APS) **によってイベントハンドリングなどが行われ，複数のアプリケーションが一つのノードで稼働できる．ま**

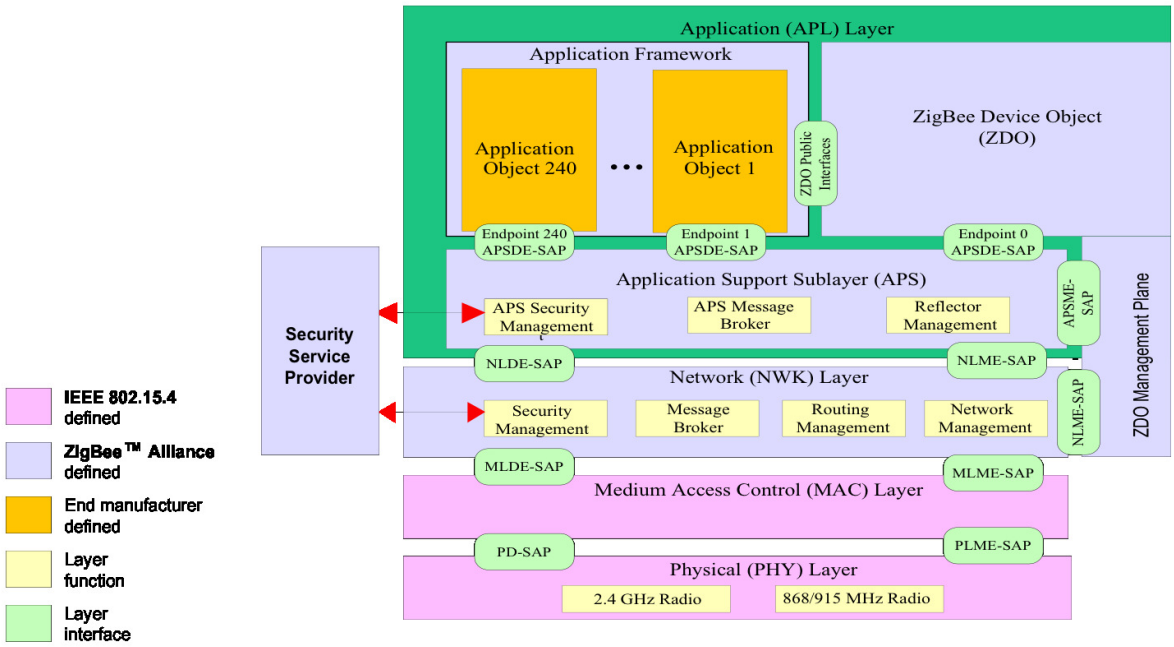


**図** 3.2: ZigBee stack

**た，**NWK Layer **以上の各層の操作をアプリケーションとして抽象化するための** ZigBee Device Object (ZDO)**，及び暗号化などのセキュリティを提供する** Security Service Provider **が存在する．**[4, p.9] **からスタックの概要を引用し図** 3.2 **に示す．**

### 3.1.2 ZigBee ネットワーク

ZigBee **ネットワークは** IEEE802.15.4 **同じく** PAN **といい，以下の** 3 **種のノードで構成される．**

**コーディネーター**
PAN **の主制御デバイスで，ネットワーク構築を担うフル機能デバイス．ルーティングをすると同時に，デバイスの** PAN **への接続管理を行う．**
**ルータ**
**マルチホップネットワークを形成し，データ (フレーム) の中継を行う．**
**エンドデバイス**
**機能の制限されたデバイスで，ルータとコーディネーターに依存して通信を行う．**

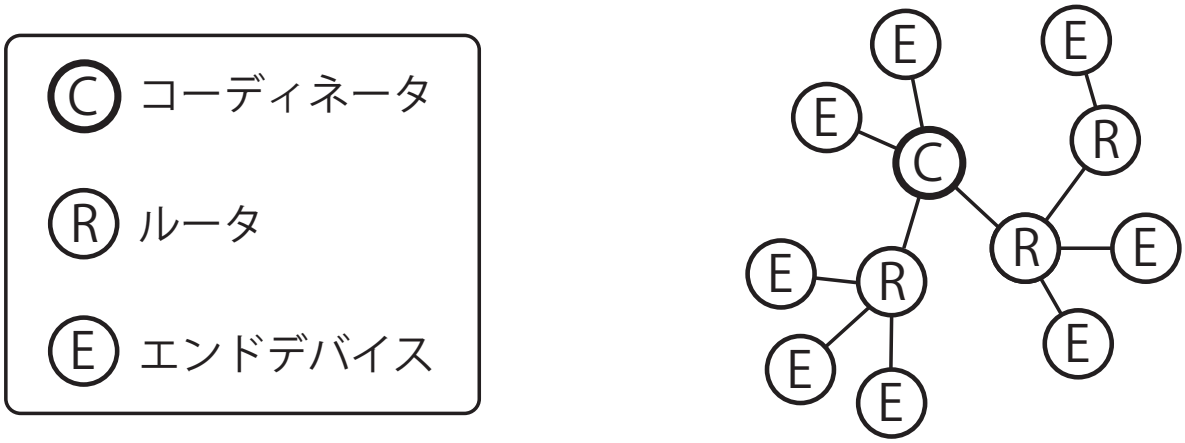


**図** 3.3: ZigBee **ノードの種類**

**また，**ZigBee **ネットワーク層はスター，ツリー，メッシュの各トポロジをサポートする．図** 3.3 **にツリートポロジを用いた** PAN **の例を示す．**

ZigBee **は、アクセスの方式として** CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) **を採用しており，隣接ノードが通信をしている場合はフレームの送信を見送る．そのため，通信ノードの増加によりフレーム送信の機会は減少する．**

### 3.1.3 ZigBee ルーティング

ZigBee (Stack Profile 0x01) **でのツリートポロジにおけるルーティングは，**NWK **アドレスの空間を分割しつつ，ルーティング情報をアドレスに含めることによって行われる．**ZigBee PAN **にはアドレス空間の分割を行い，結果的にツリートポロジの制限をするパラメータが存在する．**

Max Depth $D$

**コーディネーターから経由できるノード数** Depth **の最大数．すなわち，コーディネーターから最も深いノードまでのホップ数．**

Max Children $C$

**各ルータに接続できるノードの最大数．**

Max Routers $R$

**各ルータに接続できるルータの最大数．**

Max End Devices $E$

**各ルータからに接続できるエンドデバイスの最大数．**Max Children **と** Max

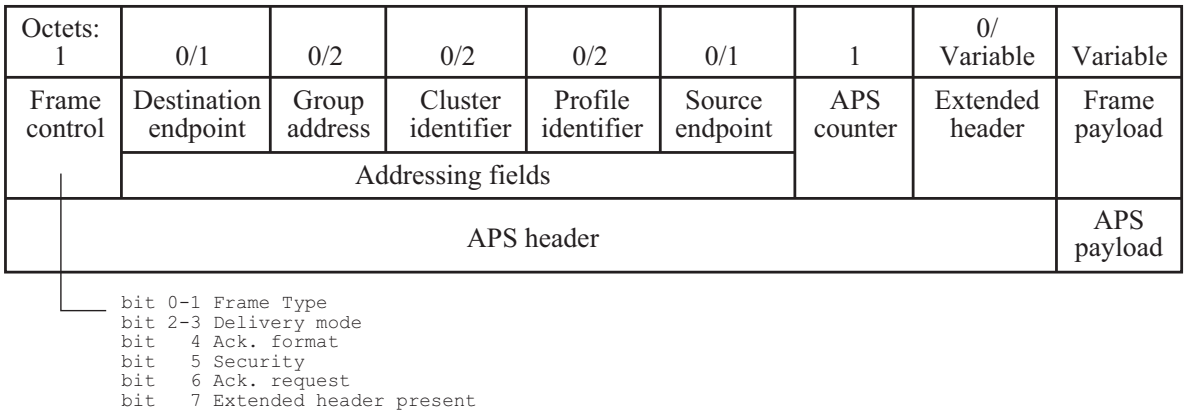


**図** 3.4: ZigBee APS **フレームフォーマット**

Routers **のパラメタから自明に導くことができる．**$E = C - R$

**ノードは自身の識別子として** NWK **アドレスを１つ使用するが，ルータの場合は親ノードからアドレスを付与される際，自身に予期される下流ノードのアドレスの最大数を同時に予約する．そのため，ルータは** NWK **アドレスのみで，フレーム宛先のノードが自身の上流か下流かを判別できる．**Depth $d$ **のルータが** Depth $d+1$ **のノードに割り当てるアドレス数を** Cskip **と呼ぶ．**Depth $n$ **における** Cskip $S_n$ **は次のように表現できる．**

$$S_D = 0$$

$$S_n = (S_{n-1})R + E$$

ZigBee **は通信メディアがワイヤレスなデバイスである．そのためツリールーティングにおいて，自分宛でないフレームを受信したルータは宛先アドレスが自己の** Cskip **の範囲内かどうかのみでルーティングを行うか判断できる．そのため，**ZigBee PAN **に存在し得る最大ノード数** $N$ **は** MAC **ショートアドレス数の** 65,535 **にはならず，次の式で求められる．**

$$N = 1 + S_0 R + E$$

### 3.1.4 ZigBee フレーム

ZigBee **では通信の信頼性向上のために** Acknkowledgement (Ack) **を利用した再送制御が用意されている．**ZigBee **で使用できる** Ack **は** MAC **層の** Ack **と** APS **層の** Ack

がある．MAC 層の Ack は MAC フレーム (図 3.1) 中の Frame Control フィールドの内の Ack Request bit を立てることによって使用できる．一方，APS 層の Ack は APS フレーム (図 3.4) 中の Frame Control フィールドの内の Ack Request bit を立てることによって使用できる．MAC 層の Ack が直接通信をしたノードから返信されるのに対し，APS 層の Ack はネットワーク層の宛先 (直接通信の宛先でも，マルチホップする宛先でも) から返信される．Ack Request が有効なフレームを送信したノードは，宛先ノードからの Ack フレームを受信するまで，もしくは Ack フレームの待受時間が経つまで，フレームのデータを破棄せずに保持する．

## 3.2 Constrained Application Protocol

IETF において利用するプロトコルとして CoAP (Constrained Application Protocol) が提案され，標準化作業が進行中である．本論文では標準化作業中文書である draft-ietf-core-coap-13 [1] を元に CoAP について述べる．

インターネットにおける Web サービスの利用は，ほとんどのアプリケーションをユビキタスへと導いた．この Web サービスは Representational State Transfer (REST) を基礎としている．

Constrained RESTful Enviroments (CoRE) は，最も制約のあるノード (例えば，RAM や ROM のサイズの制限された 8-bit マイコン) や，ネットワーク (例えば，6LoWPAN[5]) においても，REST アーキテクチャを実現することを目的とする．6LowPAN のような制約のあるネットワークでは，IPv6 のパケットを小さいリンクレイヤーのフレームに高度に分割できる．CoAP の目標の一つはメッセージのオーバーヘッドを小さいままにし，分割を抑えることである．

CoAP の主たる目標の一つは，エネルギーやビルディングオートメーション，その他の machine-to-machine (M2M) のアプリケーションのような制約環境下に特化した要求を満たす，一般的な Web プロトコルを考案することである．CoAP のゴールは HTTP の圧縮を隠蔽することではなく，HTTP と共通ではあるが，M2M アプリケーションに効果的な REST のサブセットを実現することである．CoAP は単純に HTTP インターフェイスの圧縮インターフェイスとしても使用できるが，より重要なのはディスカバリーの内蔵やマルチキャストのサポート，非同期通信など M2M のための機能を備えていることである．

CoAP は以下の主要な機能を備えている．

- M2M の要求を満たす制約 Web プロトコル
- 信頼性担保を選べるユニキャストとマルチキャストを備える UDP
- 非同期通信
- ヘッダーのオーバーヘッドが少なく，解釈しやすい
- URI と Content-type のサポート
- 単純な proxy とキャッシュの機能
- ステートレスな HTTP マッピング*1
- Datagram Transport Layer Security (DTLS) に基づいたセキュリティ

*1: proxy は HTTP を通じて CoAP リソースへのアクセスを提供したり，HTTP のインターフェイスのために CoAP の代理になっても良い．

### 3.2.1 通信概要

CoAP は制約のあるネットワークでの通信を想定しており，通信は基本的に UDP で行われる．信頼性を必要とする通信のためにメッセージタイプが設定されており，後述のメッセージヘッダの Type を confirmable に指定することで Acknkowlegement (ACK) を用いた通信が可能である．

### 3.2.2 メッセージフォーマット

CoAP は図 3.5 に示すメッセージフォーマットを取る．

Version

2bit の符号無し整数型で表される，CoAP のバージョン番号であり，この指定は

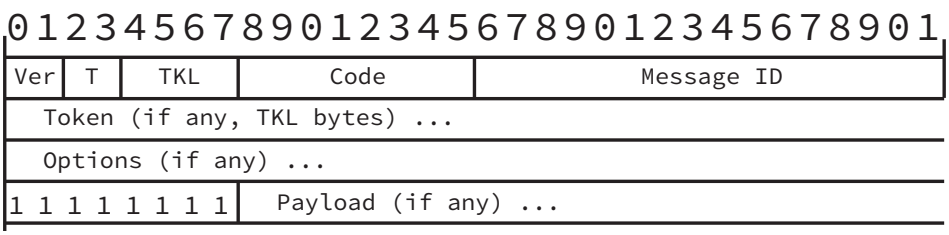


**図** 3.5: CoAP **のメッセージフォーマット**

**必ず** 1 **にする必要がある．他の値は今後のバージョンのために予約されている．**

Type (T)

1bit **の符号無し整数型で表される．メッセージは** Confirmable (0)**，**Non-Confirmable (1)**，**Acknowledgement (2)**，または** Reset (3) **のいずれかのタイプとして区別される．**

Token Length (TKL)

4bit **の符号無し整数型で表され，**Token field **の** 0 **から** 8 bit **の間を示す．**9 **から** 15 bit **の間は予約されており，この部分を埋めたメッセージを送信してはならず，埋まっているメッセージはフォーマットが間違っていると判断する必要がある．**

Code

8bit **の符号無し整数型で表される．メッセージの中身が，リクエスト** (1-31) **なのか，レスポンス** (64-191) **なのか，空** (0) **なのかを区別する．**

Message ID

16 bit **の符号無し整数型で表される．メッセージの重複検知や，**Acknowledgement / Reset **型のメッセージと** Confirmable / Non-confirmable **型のメッセージとの対応関係確認に使用する．**

Option

Option **については後述する．**

### 3.2.3 Option

CoAP **ではメッセージに含められるオプションがいくつか定義されている．メッセージに含まれるオプションインスタンスにはオプション番号，オプションのデータ長そしてオプション値そのものが含まれる．後述の** Proxy **を利用する時などには，この** Option **領域に指定形式に添って値を入れる．**

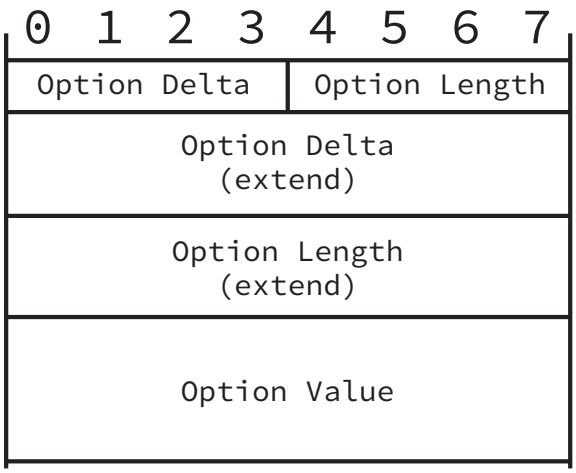


**図** 3.6: CoAP **のオプションフォーマット**

## 3.3 Proxy

CoAP **ではシステムの性能向上のためにプロキシが活用できる．**CoAP **におけるプロキシはエンドポイントの一つであり，**CoAP **クライアントの代理として動作する．例えばセンサーデバイスがリクエストへの応答ができない場合に，プロキシは保持しているキャッシュからセンサーデバイスの代わりに応答でき，応答時間やネットワーク帯域や電力消費を低減できる**.

**制約のある** RESTful **環境ではプロキシは別の役割を果たす．クライアントによって選択されるプロキシは** ”forward-proxy” **と呼ばれる．一方，本来のサーバの代わりを果たすプロキシは** ”reverse-proxy” **と呼ばれる．また，**CoAP **リクエストを** CoAP **リクエストへと対応づける** CoAP-to-CoAP **プロキシと，**HTTP **など他のプロトコルと対応づける** cross-proxy **は必要とされる機能が異なる．本論文で単に** proxy **と行った場合は，**CoAP-to-CoAP proxy **を指す．**

**プロキシは一般的に受信したリクエストに基づいて宛先を決定する機能が必要とされる．この方法は** forward-proxy **では指定されているが，**reverse-proxy **では指定されておらず各自で機能を決めることとなる．**

## 3.4 Observer

CoAP サーバー上のリソースの状態，つまりセンサーデバイスが取得するセンサー情報は常に変化し続ける．アプリケーションによってはこの変化を継続的に観測する手段が必要とされる．リソースの変化を継続観察するためのプロトコルの拡張機能が，Hartkes によって作業中文章 [6] として提唱されており，本論文ではこの作業中文章を元に CoAP observe について述べる．

この拡張機能，observe によりサーバーはリソースの状態を知りたいクライアントに対して，リソース状態通知のための継続的な応答が可能となる．このプロトコルは，リソースの最新状態をクライアントに送信する際にはベストエフォートアプローチに従い，それぞれのクライアントが持つリソースの状態と，実際のリソースの状態の結果整合性を提供する．

REST のコミュニケーションモデルとは，クライアントとサーバーがを名前空間による表現によって，リソースの状態について情報交換を行うモデルである．サーバーはその名前空間においてリソースの源として表現される．リソースの状態を知りたいクライアントはリクエストをサーバーに送信する．これに対し，サーバーはリクエストを受けた時点リソースの状態を返す．

しかし，このモデルは，クライアントがリソースの状態を継続的に知りたい場合にはうまくいかないことで知られている．この問題に対する Repeated polling or long-polls (RFC6202) のような HTTP 上で用いられているアプローチは，処理の複雑さや，オーバーヘッドのため CoAP の想定する制約環境下には不向きである．

ここで述べる CoAP の拡張では，REST のプロパティを保持したままで，一定期間ごとにサーバーからクライアントへとリソース状態を繰り返し応答する手法が提案されている．この手法の目的は，HTTP で用いられる既存の手法が解決している全ての問題点を解決することや，より全般的に問題を解決する publish/subscribe ネットワークの代わりになることではない．

提案手法のプロトコルは observer デザインパターンに基づいている．このデザインパターンでは，observer と呼ばれる要素は subject と呼ばれる既知の情報提供者に対し，subject の状態変化を通知するために登録リクエストを出す．登録リクエストを受けた subject は登録を受理した observer のリストを管理しなくてはならない．もし複数の subject の状態を知りたいのであれば，observer はそれぞれの subject に対して別々に登録する必要がある．このような observer デザインパターンは，データストレージとユー

ザーインターフェイスを分ける時のように，関連する複数の要素を完全に分ける必要がある時に使用される．CoAP では，このデザインパターンを以下のように実現する．

Subject
: CoAP において，subject は CoAP サーバーの名前空間内のリソースの一つである．リソースの状態は不定期なアップデートから連続的な遷移まで，様々な幅の時間経過で変化する．

Observer
: Observer はリソースの状態を継続的に知りたい CoAP クライアントである．

Registration (登録)
: クライアントは，拡張された GET リクエストをサーバーに向けて送信することで subject への登録を行う．このリクエストを受信したサーバーは単に目的とされるリソースの状態を通知するだけではなく，そのリソースの observer リストにクライアントを追加する．

Notification (通知)
: リソースの状態が変化すると，サーバーはそのリソースの observer リストに登録されている各クライアントに対して通知する．各通知は登録時の GET リクエストに対する返事として，サーバーから送られる追加の応答であり，リソースの最新状態の情報を含む．

図 3.7 に CoAP クライアントがリソースに登録し，3 回の通知を受ける例を示す．1 回目の通知は登録直後の状態の通知であり，2，3 回目の通知はリソースの状態が変化した時の通知である．登録リクエストと通知は Observe オプションによってその他の通信と判別される．全ての通知は，クライアントによってリクエスト内に定められたトークン (token) の値を含む．これのトークンによって，クライアントは各通知がどのリクエストと対応しているかを判断できる．

クライアントがリソースの状態を知りたいとサーバーが判断する限りそのクライアントは observer リストに残り続ける．サーバーはリソースへのリクエストが継続しているかを，クライアントが送信する Acknowlegement によって判断する．クライアントがサーバーからの通知を拒否したり，もしくは Acknowlegement が失敗した場合には，サーバーはクライアントがリクエストを継続を望んでいないと判断し，そのクライアントは observer リストから削除される．

通知は他の応答と同様に保存が可能であり，次の新しい通知が現れるまでは最新の情報として利用できる．各通知はサーバーが次の通知を遅くともいつまでに送るか

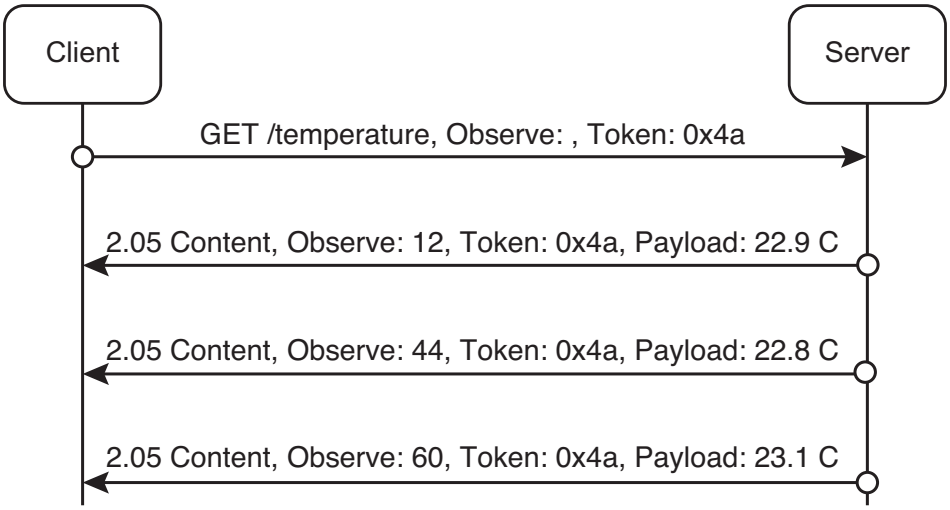


**図** 3.7: CoAP observe **のシーケンス**

(Max-Age) **を含む．これにより通知をしばらく受けていないクライアントは，単純にリソースがまだ状態変化をしていないのか，または次の通知が期限を超えて遅れているのかを判断できる．同様にサーバーはクライアントがリソースへのリクエストを継続しないと判断するのにも利用できる．クライアントがリクエストの継続解消後に通知を受けたい場合は，もう一度登録作業が必要となる．**

Observe **通信においてプロキシを使用する場合のシーケンスは図** 3.8 **のようになる．プロキシはクライアントからの** Observe GET **コマンドを受信すると，**Proxy-uri **から次の宛先サーバーのアドレスを導き，新規トークンを含めた** Observe GET **コマンドをそのサーバーへ送信する．プロキシはこの転送処理を行う際に扱う，クライアントとプロキシの間で利用されるトークンと，プロキシとサーバーの間で利用されるトークンの対応関係を保存する．プロキシこの対応関係を利用して，サーバーからの定期応答を適切なクライアントへと転送する．**

## 3.5 本章のまとめ

**本章では，**IoT **システムの基盤技術であるワイヤレスセンサーネットワーク技術である，**IEEE 802.15.4**，**ZigBee **について述べ，また，**Constrained Network **を含む複数のアクセスネットワークにまたがって** End-to-End **通信を行うための技術である** CoAP **について述べた．**

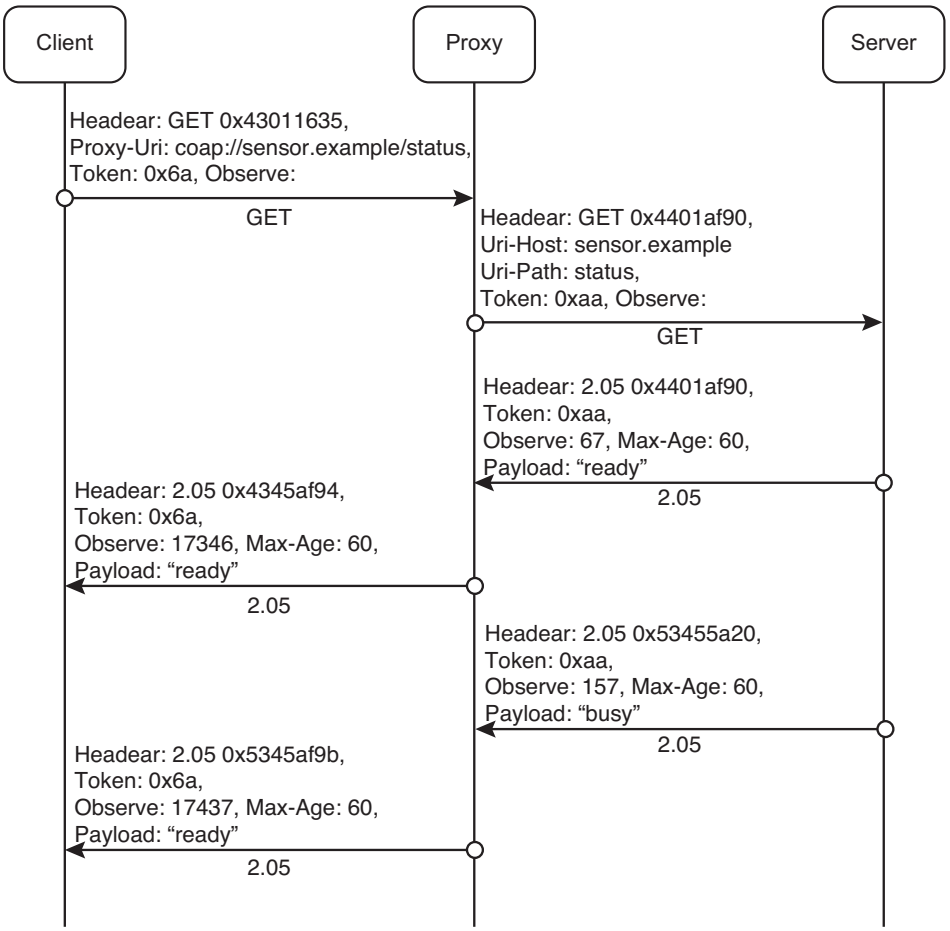


**図** 3.8: CoAP observe **のシーケンス**

# 第 4 章

# ワイヤレスセンサーネットワークの輻輳

本章では，まずワイヤレスセンサーネットワークにおける通信量について考察し，シミュレーションによって理論的な通信量の増加について述べる．また，実測実験による調査からワイヤレスセンサーネットワーク上での輻輳について考察する．

## 4.1 ワイヤレスセンサーネットワークの通信量

センサーネットワークにおける通信量は，以下の要素に依存する．

- 通信ノード数
- トポロジー
- ノードの通信頻度
- データサイズ

**通信ノード数**

ワイヤレスセンサーネットワーク上での通信は，所属するノード間の通信であり，ネットワーク全体の通信量はノード数の影響を受ける．

**トポロジー**

ワイヤレスセンサーネットワークは，目的に応じてそのネットワークのトポロジーを選択できる．スター，メッシュ，クラスタツリーなどの種類が存在するが，ここではクラスタツリートポロジーについて言及する．クラスタツリートポロジーでは通信量はツリーの構成状況に左右される．外部ネットワークとの接続点がコーディ

ネーターだと仮定した場合，子ノードの通信は親ノードを中継する．あるセンサーノードが与える通信量はそのノードがネットワークにおいて，どのツリー層に存在するかによる．例えば，0 層をツリーの根であるコーディネーターとすると，3 層目に存在するエンドデバイスはコーディネーターに対してデータを送信する場合には，到達までの間に 2 つのルータノードを中継する．そのため，1 つの通信に 3 パケットが必要となる．

**ノードの通信頻度**

センサーノードの通信頻度はクライアントの数と各クライアントの要求の性質に影響される．クライアントからの要求が操作など単発のものであれば，クライアントが要求を出す間隔がそのままセンサーノードの通信頻度となる．一方，CoAP の observe コマンドのように一定時間ごとにリソースの状態を送信する場合には，クライアント数とそれらが指定する応答間隔が通信頻度に影響する．

**データサイズ**

やりとりをするデータのサイズが一つのパケットに収まらない場合には，複数のパケットに分割して送信する．例えば，IEEE 802.15.4 の一つのパケットのサイズは 127 byte であり，そのうち 100 byte が APS ペイロードとして使用可能である．そのため，100 byte よりも大きいデータを送信する際にはパケットの分割が必要となる．

### 4.1.1 シミュレーション

前述の通信量への影響要素のいくつかを変化させることにより，通信総量がどのように変化するかのシミュレーションを行った通信総量の単位は秒間当たりの発生パケット数 (packet/sec) とした．要素の設定をまとめて表 4.1 に示す．

まずノード数については，一般世帯での使用をメインに考察を進める．一般世帯に存在する家電製品全てにセンサーモジュールが組み込まれていると仮定した場合，家電の数がセンサーノードの数となる．日本において一般世帯が所有する家電の数は 50 程度である．センサーネットワークに参加する家電の数が少ない場合と，一般家庭の持つ家電が増加した場合について考慮し，1 台から 100 台までの間で考察する．次に応答頻度については，1, 10, 30, 60 秒に一度とした．ツリーの構造は，同じ層のノード全てが限界まで子ノードを持った時に，次の層のノードが子ノードを持つと仮定する．各ノードが保持できるノード数は 2 から 5 とした．

表 4.1: シミュレーション条件

| 要素 | 変動幅 | 設定値 |
|---|---|---|
| ノード数 | 1〜100 | 50 |
| 応答間隔 | 1 回/1, 10, 30, 60 秒 | 1 回/30 秒 |
| クライアント数 | 1〜10 | 1 |
| トポロジー | クラスタツリー | - |
| クラスタごとの子ノードの数 | 2〜5 | 4 |

センサーネットワークのトポロジーについて考察する．家庭全体にセンサーネットワークを行き渡らせることを想定した場合，クラスターツリーネットワークを選択する．

センサーノードの通信頻度について考察する．センサーノードの通信は，コマンドによるノードの制御などの一度きりの通信と，センサーデータを定期的に送信する通信とに分けられる．センサーデータ収集を目的とするシステムを想定している本研究においては，後者が通信の大半を占めることとなる．定期的な通信の頻度はセンサーデータの収集間隔による．

### 4.1.2 シミュレーション結果

以上の条件を元にしたシミュレーションの結果を以下に示す．

まず，ツリーの構成が与える影響について考察した．ツリーの各ノードが持てる子ノードの数を 2 から 5 の間で変化させると，図 4.1 のような結果を得る．この図から持てる子ノードの数を多くすることにより，メッセージが中継される機会が減少し，ネットワーク全体の通信量を低減できることがわかる．しかし，増加にともない低減の効果が減少することも分かる．

次にクライアント数とパケット数の関係について考察した．各ノードが持てる子ノードの数を子ノードの数は 4 とし，また，センサーノードの応答間隔を 1 分に 1 回とした場合に，クライアント数とノード数，秒間発生パケット数がどのように動くかを図 4.2 に示す．クライアント数の増加と発生パケット数の増加は比例することが分かる．

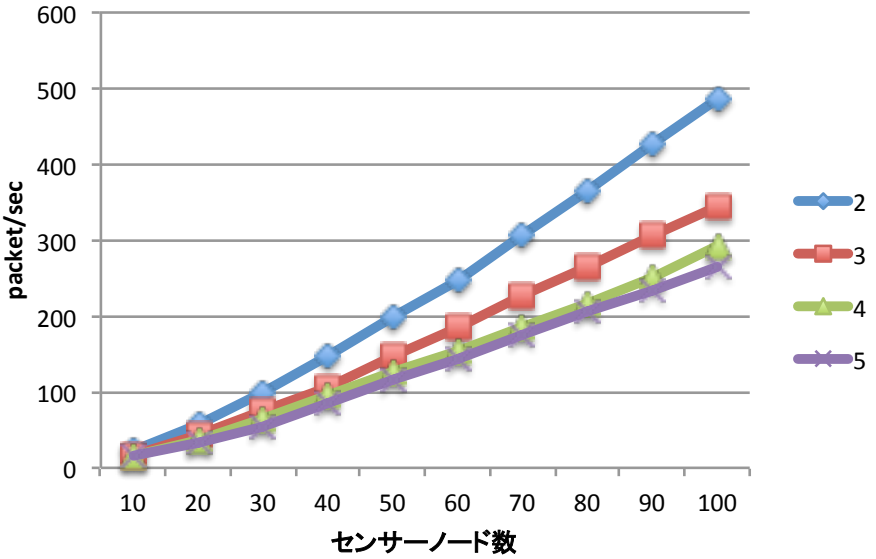


**図** 4.1: **ツリーの子ノード数とパケット数の関係**

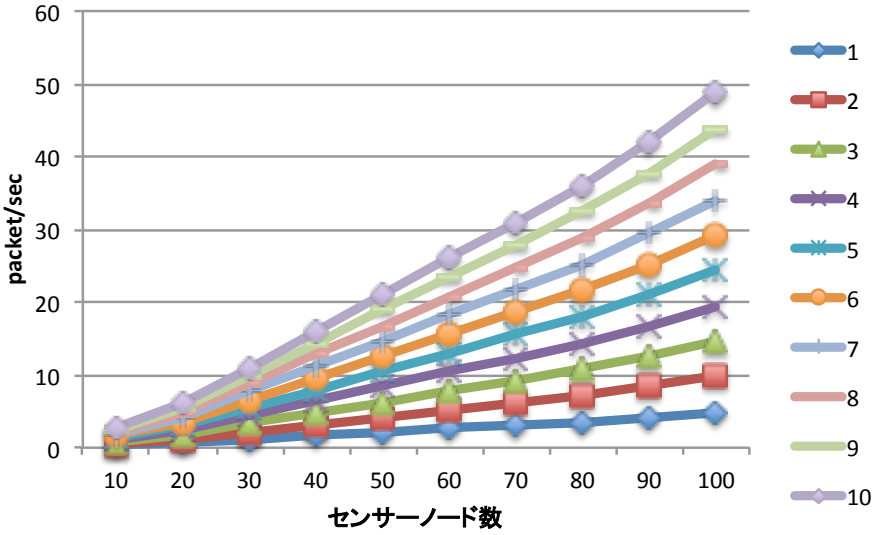


**図** 4.2: **クライアント数とパケット数の関係**

## 4.2 輻輳の測定

実際のワイヤレス環境において通信状況がどのように変化し，どのタイミングで輻輳が発生するのかを確認するために，実機を用いた実験を行った．実験には第 2.3.1 節で示した環境を用いた．

### 4.2.1 APS の RTT

まず始めに ZigBee 通信機の Application Sublayer (APS) のラウンドトリップタイムについての測定を行った．1 つのセンサーデバイス，1 つのルーター，1 つのコーディネーターから成る 2 ホップのネットワークを形成し，同チャネルでの ZigBee デバイスの通信が無い環境下において実験を行った．センサーデバイスは継続的に固定長のデータをコーディネーターに送信し，コーディネーターは APS Acknowledgement (APS_ACK) を返信する．この通信状態を Daintree 社の Sensor Network Analyzer (SNA) を用いて観測した．

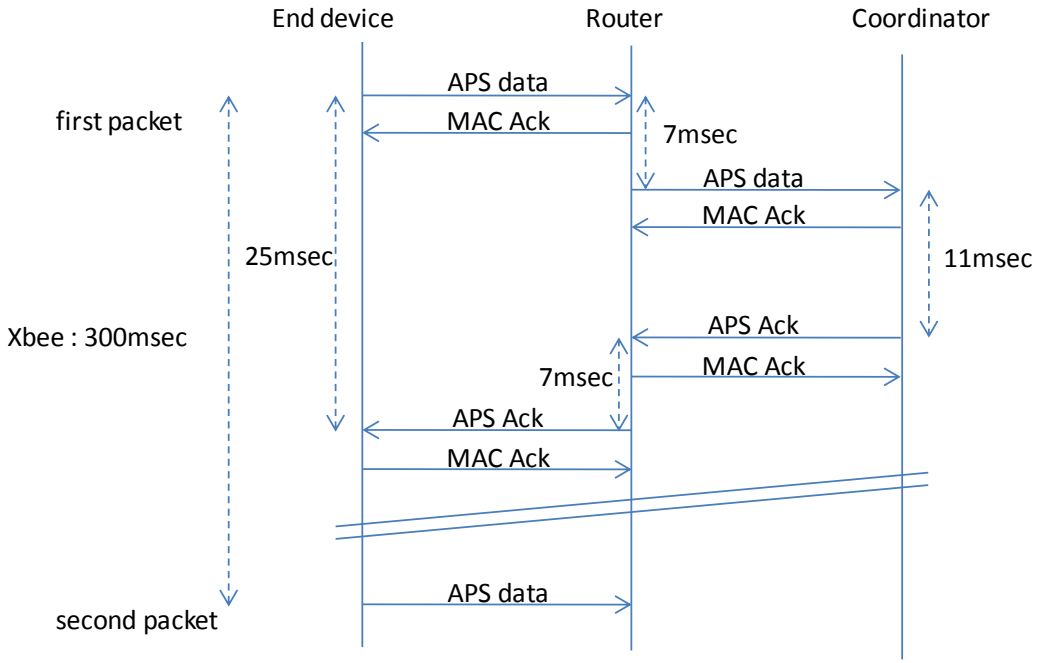


図 4.3: APS 通信のラウンドトリップタイム

図 4.3 に計測結果を示す．ワイヤレス区間での経過時間は無いとするとルーターが転送

処理を行うのにかかる時間は，ルーターへのパケットが送信されてから，ルーターが転送パケットを送信するまでの時間であり， 7 msec であった．また，コーディネーターが APS データを受信してから APS_ACK を送信する時間は，同様に考えると 11 msec だった．

### 4.2.2 センサーデバイスの処理速度

次に，センサーデバイスの送信間隔を 300msec から 1000msec の間で変化させ，ワイヤレス区間での送信時間の計測を行った．センサーデバイスからコーディネーターへは 100 パケットを送信した．ワイヤレス区間における時間計測には SNA を用いた．計測結果を図 4.4 に示す．

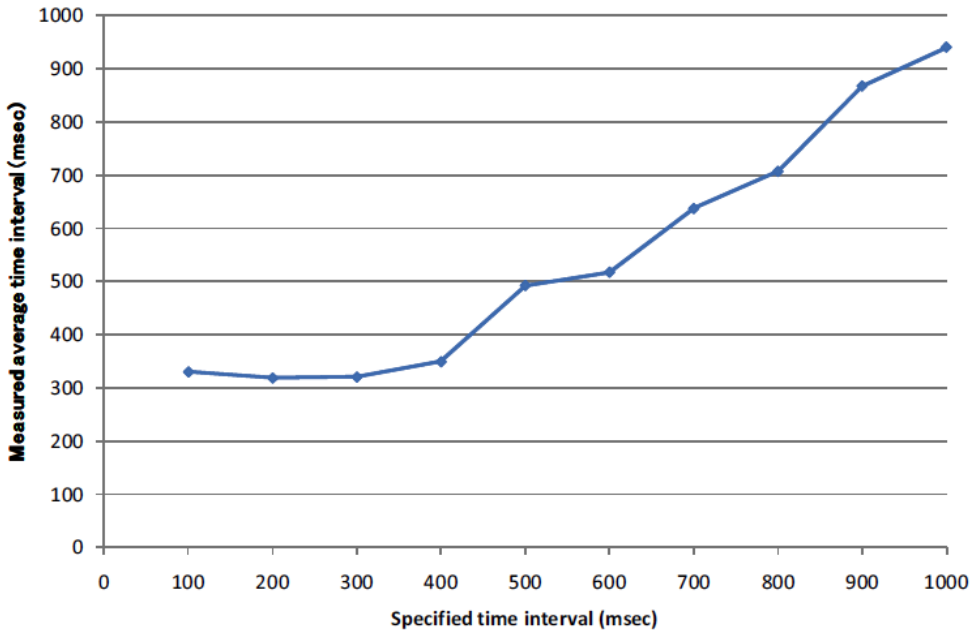


図 4.4: ワイヤレス通信の送信時間

センサーデバイスは 300msec 以下で送信できないと分かる．もし，これよりも短い間隔で強制的にパケットを送信しようすると，パケットはトランシーバーの内部の送信キューに蓄積される．MSP430 内の通信コントロールファームウェアよると，9600 bps の UART を通じて XBee で送る実装において APS データを送信するのには，全体で最小 300 msec がかかる．理論的な最小間隔を考えると，100 byte のデータは UART 区間

ではスタートビットとストップビットを考慮して，$104msec(= 100 \times 10/9600)$ となる．UART 通信は仮想的にフロー制御を行っており，UART 通信区間において 300 msec かかるというのは妥当である．

以上に連続した２パケットの間隔は 300 msec であると示した．このことから，一つのエンドノードから送信する最短間隔を 200 msec のマージンを含めて 500 msec とした．これは同一のセンサーデバイスから生成される連続した２つの APS パケット間に関する，APS 通信の値である．APS のペイロードの最大値は 100 byte であるため，一つのセンサーデバイスから得られる APS 通信の最大速度は $1.6kvps(100 \times 8/0.5)$ となる．

同様に ZigBee ルーターにおける転送速度についての測定も行った．測定からパケットを転送するのには 7 msec かかることがわかった．つまりセンサーモジュールからセンサーデータを取得して送信するよりも，$115kbps(100Byte \times 8bit/7msec)$ だけ転送のほうが早く処理を行えることがわかる．

### 4.2.3 センサーネットワークの通信速度

各センサーデバイスの送信間隔をトランシーバーの過負荷を防ぐため 500 msec とし，送信センサーデバイスの数を 5 分につき 1 つずつ増加させてネットワークの輻輳を計測した．センサーデバイスはエンドデバイス，ルーター共に通信経路はルートディスカバリーの結果に依存する．実験は研究室の一画にて行った．図 4.5 に示すように，$(612m^2 = 34m \times 18m)$ の作業室内にコーディネーター，全てのセンサーデバイスを共に設置し，4 つのルーターを中間地点に配置することでマルチホップネットワークを形成した．

上記のネットワークにおけるセンサーデバイスの APS スループットは，Postgres データベースに蓄積された受信パケットの時間間隔をから算出できる．APS スループットの総計は全ての APS スループットとホップ数から導ける．全体で 55 ホップする 20 のエンドデバイスの通信について平均して 60 kbps のスループットを得た．センサーデバイスの数が 14 になるまでは，センサーデバイスの数とスループットはの関係は線形である．これは ZigBee ネットワークにおいて輻輳が発生していないことを示している．各センサーデバイスのスループット限界である 1600 bps から，ホップ数とネットワーク全体のスループットの関係を計算し，上記の実験の測定結果を比較し図 4.6 に示す．

図から送信センサーデバイスの数が 16 までの状態では，理論値と同等の性能を出していることが分かる．理論値との差はセンサー MCU とトランシーバーの間のシリアル通信に影響される．

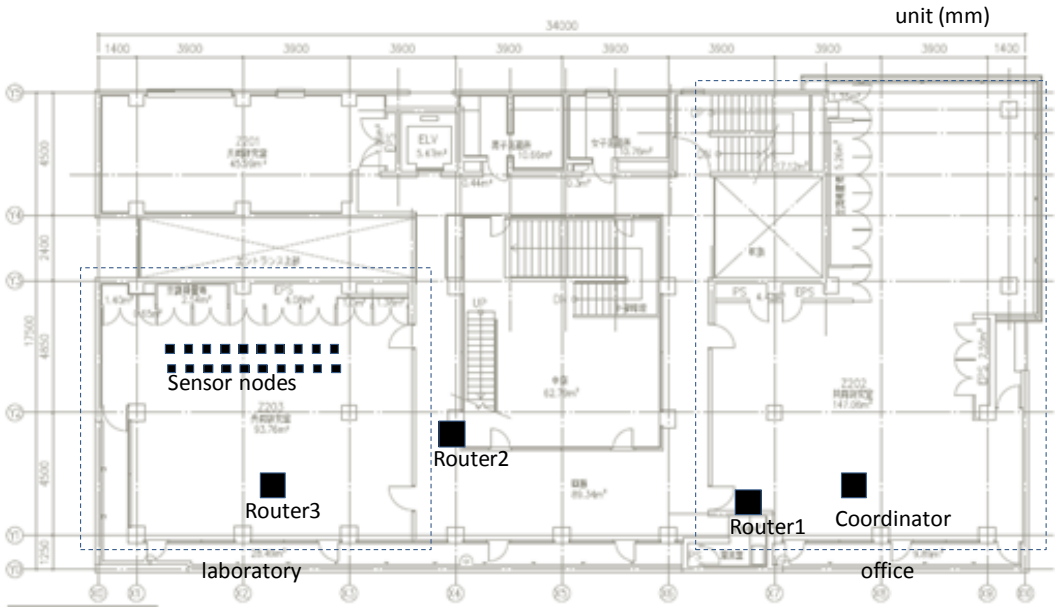


**図 4.5: 実験時のセンサー配置**

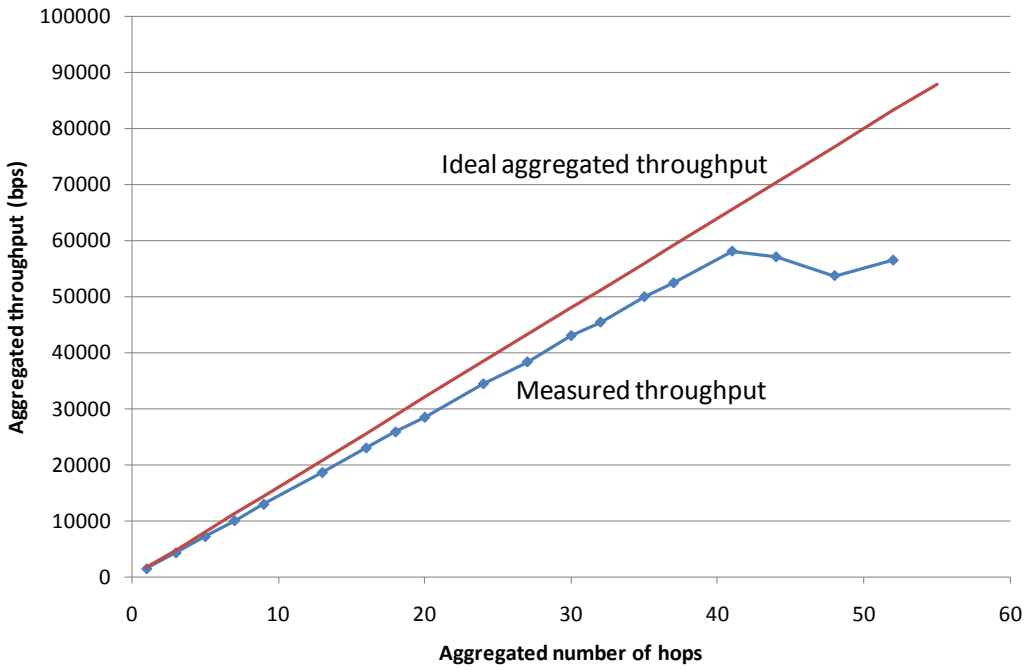


**図 4.6: センサーネットワークの通信速度**

### 4.2.4 定期ポーリングによる輻輳

**定期的なポーリング要求をゲートウェイを中継してセンサーデバイスへ送り，その通信状況からネットワークの輻輳状態を測定した．ポーリングの間隔は** 100 msec **から** 1000 msec **とし，各送信タイミングごとに** 100 **ずつポーリング要求を出した．要求に対する成功応答，センサーデバイスによって破棄された要求の数，ゲートウェイによって破棄された要求の数を計測することで，ネットワークのどの地点で輻輳が発生しているのかを測定する．**

**測定結果を図** 4.7 **に示す．ポーリング間隔が** 300 msec **以下の場合には，ポーリング要求の一部はゲートウェイによって破棄された．ポーリング間隔が** 700 msec **以下の場合については，一部の要求はセンサーデバイスによって破棄された．これはセンサーデバイスへの過負荷が原因と予測できる．この実験における通信量は** 1.1 kbps $(= 100byte \times 8bit/0.7sec)$ **であり，上述した** ZigBee **ネットワーク全体の通信容量** 60 kbps **よりも十分に小さい．よって，**ZigBee **ネットワーク全体としての問題は無い．**

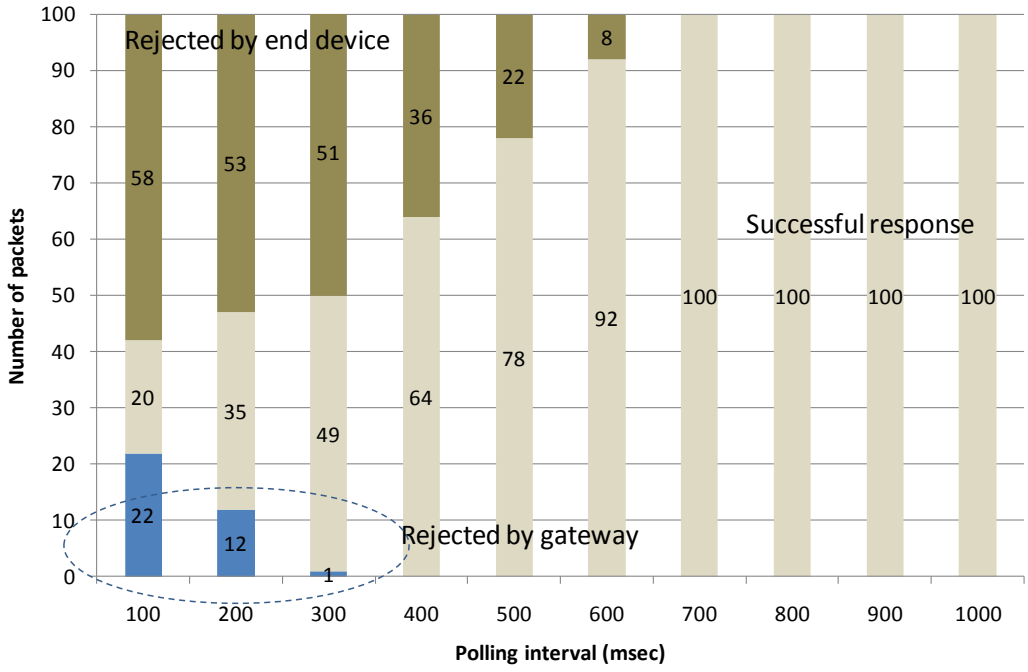


**図** 4.7: **定期ポーリング測定結果**

## 4.3 本章のまとめ

本章では，まずワイヤレスセンサーネットワークにおける通信量について考察し，シミュレーションによってワイヤレスセンサーネットワークにおける通信量の変化について述べた．また，実環境における測定によりワイヤレスセンサーネットワーク上での輻輳について考察した．

# 第 5 章

# 集約技術

本章では，ワイヤレスセンサーネットワークにおける集約技術について述べ，関連する研究を示す．

## 5.1 ワイヤレスセンサーネットワークにおける集約

ワイヤレスセンサーネットワークにおいて，輻輳を軽減するための手法として集約技術がある．集約技術はマルチホップネットワークにおける情報の収集やルーティング，中継ノードでのデータ処理を減らすことで，消費電力を主とするリソース消費の抑制，ひいてはネットワークの寿命を延ばすことを目的とする．[7] では以下のように集約技術の分類，比較が述べられている．

まず，集約技術は集約するデータの扱い方から以下の 2 つに分けられる．

**サイズの変更を行う**

複数のソースから来るデータを結合や圧縮することで，ネットワークを通じて送信するデータの量を減らす．例えば，二つのソースから得た温度を二つのパケットに分けて送らずに，一つのパケットにまとめて送ることや，複数の取得センサーデータの平均を計算して一つのパケットで送るなどがこれにあたる．

**サイズの変更を行わない**

複数種類のデータを一つのパケットで送る．これらのデータは，合わせて一つのデータとして処理することはできないが，ひとつのパケットには入る場合に適用でき，送信パケット数を減らすことにより，オーバーヘッドを軽減する．

前者の手法は，ネットワーク内の通信量を低減するのには有効であるが，収集したデー

タから得られる情報の正確性が損なわれる可能性がある．これはデータの集約が可逆，非可逆のどちらで行われたかに依存する．一方，後者の手法は元のデータを完全に維持するため情報の正確性は担保される．こちらの手法では，集約の効果はアプリケーションやデータの収集頻度，ネットワークの特性などの要因に効果が左右される．両手法共にデータ集約には異なるレイヤーにまたがって処理が必要となる．

また，集約技術は 3 つの基本的な特徴で分類できる．

- ルーティングプロトコルへの変更
- 集約手法
- データの再現

### 5.1.1 ルーティングプロトコルの変更

データ集約技術の中にはルーティングプロトコルに変更を加えるものがある．古典的なルーティングプロトコルは典型的には目的地までの最短経路をたどる．しかしデータ集約を考慮した場合には，ノードはパケットの中身を考慮し，より集約が可能となるパスを選択する必要がある．このような転送はデータセントリックルーティングと呼ばれる．データセントリックルーティングでは，ノードはデータ集約の実施点やデータタイプ，情報の優先度などを考慮した上で，関連するノードの情報を検索する．また，アプリケーションの特性やルーティング構造，データ集約手法に適したルーティングを行う必要がある．さらに，センサーネットワークにおける集約では，ノード間の同期についても考慮する必要がある．取得したデータを即座に送信することは，全ての場合において最良の選択とは限らない．近隣のノードからの情報を待つことでデータ集約を行う機会が生まれ，結果としてネットワークの性能改善をもたらすこともある．このようなデータ送信のタイミングについての配慮は，センサーノードが取得した情報を定期的にシンクノードに送信する観測アプリケーションにおいて必要とされる．[8] では，タイミングの調整方法は以下のようにまとめられると述べられている．

**定期送信**

この手法では各ノードは事前に定めた期間待機し，そのタイムアウトまでに取得し

たデータを集約して送信する．

近隣ノードとの調整

この手法では，ノードは全ての子ノードからデータを得たタイミングで取得データを集約して送信する．各ノードは自身に属する子ノードを把握する必要がある．また，子ノードの離脱や子ノードのパケットがロストした場合を考慮し，タイムアウト処理が必要となる．

定期的なホップごとの集約調整

この手法では，ノードはデータ収集のツリーの中で自身がどの位置にいるかを把握し，その位置によってタイムアウトのタイミングを決定する．そのタイムアウトまでに得たデータについて集約して送信する．この手法では，タイミングの決定はルーティングプロトコルの設計に依存する．[9] ではルーティングプロトコルとタイミングについて述べている．

### 5.1.2 集約手法

集約技術において重要な特徴のひとつにデータの集約手法がある．集約手法には様々な種類があり，これらはほとんどの場合にはアプリケーションの特性と密接に関係している．集約手法は以下の観点から分類できる．

可逆性

集約手法は非可逆，可逆のいずれかによってデータの圧縮または結合を行う．非可逆の集約手法では，集約データから元の情報を完璧に復元することはできない．つまり，収集するデータの正確性が損なわれる可能性がある．一方，可逆手法では元の情報を保ったまま集約する．これは，データの受信側で集約データから完全な情報が復元できる手法を示す．

重複への考慮

ノードは同じ情報の複製を複数回扱う可能性がある．情報を集約する際に同じデータが複数回処理される場合がこれにあたる．集約手法が重複について考慮しているのであれば，同じデータを扱う回数が手法のもたらす効果に影響する．一方で集約手法が重複を考慮しない場合では，データの重複は手法がもたらす効果に影響を与えない．例えば，いくつかのデータの平均値を使用する手法はデータの重複に影響を受けるが，最小値を使用する手法はデータの重複にあまり影響を受けない．

ワイヤレスセンサーネットワークに適した集約手法を考えると，他の観点も浮かび上がる．ワイヤレスセンサーネットワークでは，計算資源や消費電力に大きく制限のあるセンサーデバイスが用いられるため，集約手法はこれらのデバイスにおいて定常的に使用できる必要がある．また，運用形態や電力供給系統，計算資源などの条件が異なる多様なデバイスにおいて，共通して利用できる必要がある．これらの観点は集約手法だけでなく，ルーティングプロトコルを設計する上での要求事項である．

### 5.1.3 データ構造

ワイヤレスセンサーネットワークのノードはストレージの制限から，受信または生成した情報を全て内部バッファに保持できない．そのため扱うデータに対して，保存，破棄，圧縮もしくは送信のいずれかの動作の選択を迫られる．これらの動作を円滑にするには情報を最適なデータ構造で表現する必要がある．どのようなデータ構造が適切であるかはアプリケーションの特性に依存する．しかし，データ構造は全てのノードで適切に利用できる必要があるのと同時に，ノード固有，設置環境固有の特性にも対応できる必要がある．

ここまでルーティングプロトコル，集約手法，そしてデータ構造という観点について個別に述べた．しかし，実際に集約技術を考える際にはこれらの観点は相関するため，全てを考慮した設計が必要となる．

## 5.2 集約手法の比較

上述したように，集約手法はデータ構造と密接な関係を持つ．最も単純な集約手法としては，Madden[10] や Lindsey ら [11] が用いられているような平均値や中央値，最小値，最大値などに加工したデータの利用がある．これらの手法は通信量を大幅に削減られるが，情報の正確性を欠くという一面もあわせ持つ．本節では，前述の分類に基づいていくつかの集約手法の比較を行い表 5.1 にまとめる．

### 5.2.1 TiNA

Temporal coherency-aware in-Network Aggregation(TiNA) [12] はセンサーネットワークにおいて，通信データが集まるツリーの根のようなノードで動作する．TiNA ではセンサーデバイスは新規取得したセンサーデータの値と前回取得した値を比較し，その差が一定以上であれば送信する．送信の判定には事前に各ノードに通知する tct の値を使用し，新規取得値 $V_{new}$ と前回取得値 $V_{old}$ の関係が以下の式を満たすかによる．

$$\frac{|V_{new} - V_{old}|}{V_{new}} > tct \tag{5.1}$$

### 5.2.2 DADMA

Data Aggregation and Dilution by Modulus Addressing (DADMA) [13] は SQL ステートメントのルールに従いデータの集約，または希薄化を行う手法である．DADMA は，取得したデータを内部に保持しているセンサーノードの集合である，ワイヤレスセンサーネットワークをリレーショナルデータベースとして扱う．センサーネットワークのデータベースとしてのとらえ方は，sensor network database view (SNDV) としてシンクノードによって決定され，管理される．DADMA の基本的な発想は，センサーグループのデータを集約することと，データ収集ツリーからいくつかのセンサーを除外することである．これらの行為は次に述べる 2 つの単純なルールに従って行われる．まず，利用者は SNDV からデータ領域のサブセットを取得でき，データは次の集約関数によって行われる．

$$f_a(x) = xdivm \tag{5.2}$$

次に，センサーノードは次の希薄化関数によりクエリー対象から除外される

$$f_d(x) = (x/r)mod(m/r) \tag{5.3}$$

### 5.2.3 AIDA

データ集約は多くの場合には，アプリケーションの特性に合わせて動作する手法が多い．しかし，Application independent data aggregation (AIDA) [14] では図 5.1 に示すように OSI 参照モデルで言うデータリンク層とネットワーク層の間に，中間層を設けることにより，アプリケーションの特性に影響されれずに集約を実行する．この中間層は，ワイヤレスセンサーネットワークの状態を動的に判断し，ネットワークの混雑状況に応じて集約を行う．AIDA で行う集約は可逆性を持つため，上位層は状況に応じて圧縮などの手法を利用可能である．

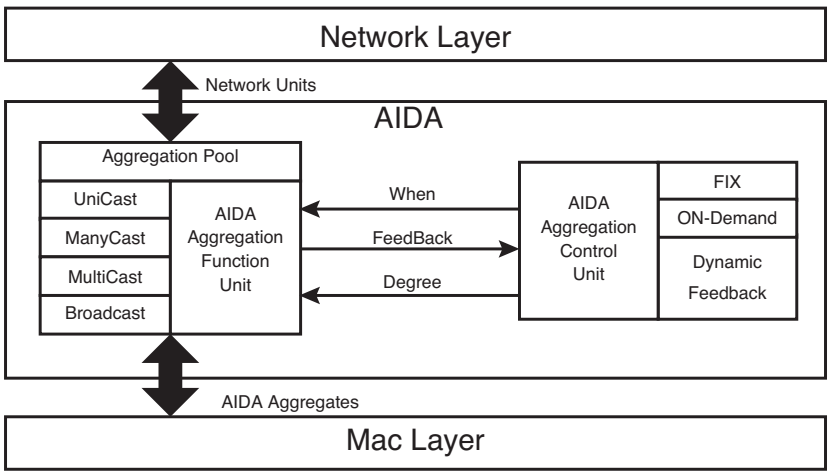


**図** 5.1: AIDA **の構成**

**表** 5.1: **集約手法の比較**

| 集約手法 | TiNA | DADMA | AIDA |
|---|---|---|---|
| 可逆性 | 可逆 | 可逆 | 可逆または非可逆 |
| 重複への考慮 | 考慮しない | 考慮する | 場合による |
| ルーティングの変更 | する | する | しない |

## 5.3 本章のまとめ

本章ではセンサーネットワークにおける輻輳を軽減する手法である集約技術について述べ，また，関連する研究についての比較を述べた．

# 第 6 章

# 応答集約

第 2 章で述べたように，多様なアクセス方式にまたがる IoT システムにおいては End-to-End 通信は重要である，しかし，第 4 章で述べたようにワイヤレスセンサーネットワークでは輻輳が問題となる．本研究では，輻輳の解決手法として応答集約に着目する．応答集約は第 5 章で述べたように，ワイヤレスセンサーネットワークにおける輻輳を軽減する手法として有効であり，様々な研究が行われている．しかし，これらの研究はワイヤレスセンサーネットワーク内での集約手法にとどまる．本章では，多様なアクセス形式にまたがる End-to-End 通信を行う IoT システムで利用できる応答集約手法を提案する．また提案手法の RACOW システム上と CoAP 上での実現手法について述べる．

## 6.1 手法概要

提案手法では，End-to-End 通信環境において複数のクライアントからのリクエストにより発生する，同一データの重複送信に着目する．まず，クライアントからのリクエストが重複しているか，つまり集約が可能かどうかは，アプリケーションの性質に依存する．End-to-End 通信ではアプリケーションデータは，通信の両端でのみ解釈が行われる．よって，応答集約の可否は複数のリクエストを受けるセンサーノードが判断する．センサーノードは同じリソースに対する複数のクライアントからのリクエストを，アプリケーションとクライアントのセッション管理データを用いて，得た応答を集約して送信する．集約応答は通信の応答集約への対応機能をもつ中継地で，このセッション管理情報を用いて各クライアントに配送される．RACOW システムにおいては IP ネットワークと ZigBee ネットワークのゲートウェイ装置に機能を追加し，CoAP においてはプロキシに機能を追加する．

本手法第 5 章で述べた集約手法の特徴を表 6.1 のように満たす．

表 6.1: 提案する応答集約の特徴

| | |
|---|---|
| 可逆性 | 可逆 |
| 重複への考慮 | する |
| ルーティングの変更 | しない |

## 6.2 RACOW システムにおける応答集約

RACOW システムにおける End-to-End 通信は，サブスクライブ POST とセンサーデバイスへのメッセージがゲートウェイ装置 (GW) において変換される際に，対応関係のリストをゲートウェイ装置が記憶しておくことで成立している．ゲートウェイ装置は両者を subscribe ID (sID) と Event ID(eID) で判別する．提案手法では，この sID と eID の対応関係を利用する．提案手法を使用した，アプリケーションとセンサーデバイスの応答集約通信は図 6.1 に示すように行われる．また，各メッセージに含まれるデータは表 6.2 に示すようになる．リクエストを受けたセンサーデバイスは集約可能なリクエストについて eID リストを作成し，記憶する．センサーデバイスは一定時間経過後に，集約可能なリクエストの eID をひとつのメッセージにまとめてセンサーデータと共に送信する．応答メッセージを受信したゲートウェイ装置は sID と eID の対応関係リストを元に，HTTP POST メッセージを作成し，HTTP クライアントとして対応する全てのアプリケーションに対して送信する．

表 6.2: メッセージ内容

| メッセージ種別 | メッセージ内容 |
|---|---|
| クライアントから GW へのリクエスト | subscrive ID, Uri-Path, dest addr |
| GW 装置からセンサーデバイスへのリクエスト | Event ID, Uri-Path |
| センサーデバイスから GW 装置への応答 (通常) | Event ID, Reply data |
| センサーデバイスから GW 装置への応答 (応答集約) | Event ID, Event ID..., Reply data |
| GW 装置からクライアントへの応答 | subscrive ID, Reply data |

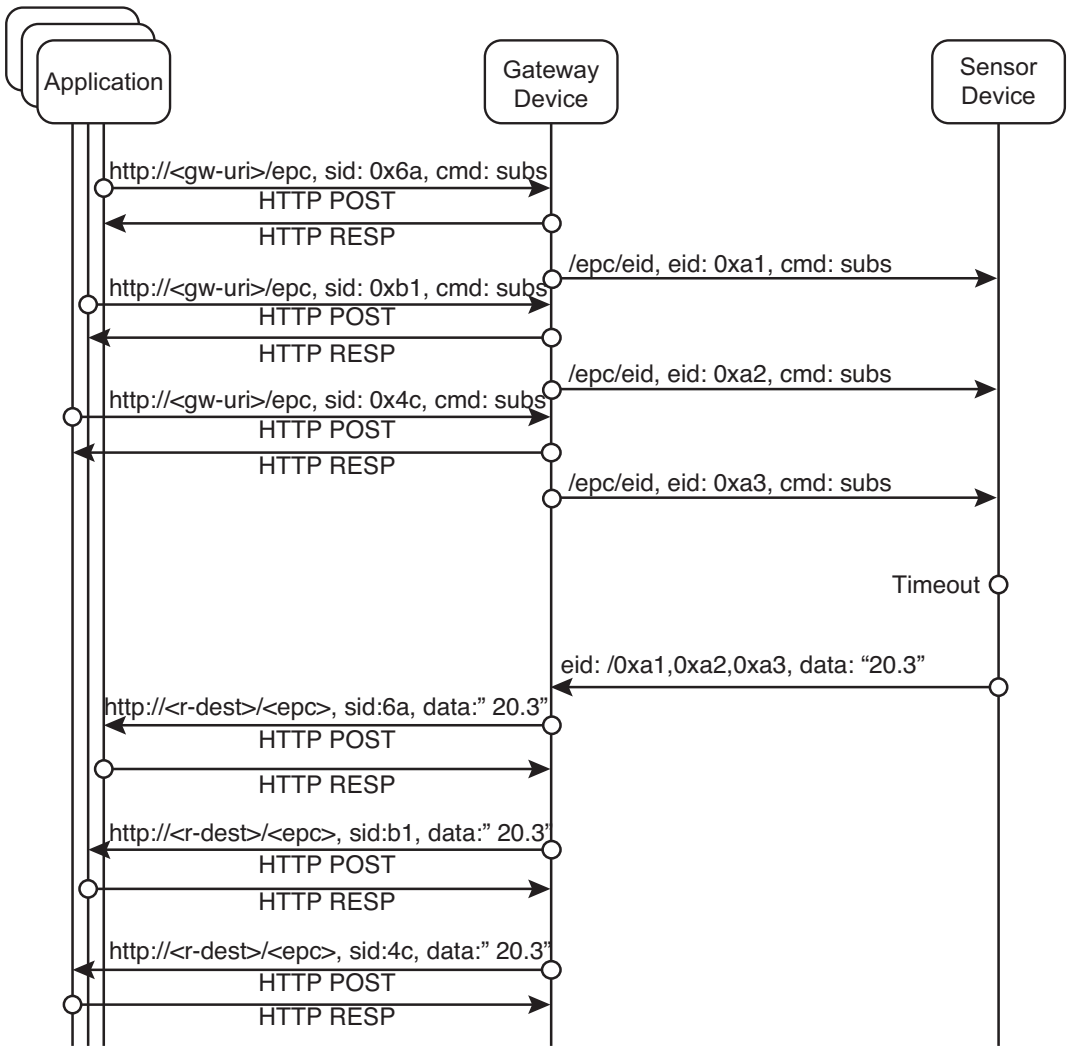


**図** 6.1: RACOW **システムでの応答集約のシーケンス**

## 6.3 CoAP での応答集約

**集約されたデータは，各クライアントへ分配する必要があるため，この処理を行う機能が輻輳を軽減したいネットワークの外に必要となる．**CoAP **では第** 3.3 **節で述べたように，システムの性能向上手段としてプロキシを提示している．このプロキシを応答集約に対応させることで輻輳の軽減を図る．プロキシは単体で用意しても，クライアントとサーバーとの通信を中継するゲートウェイ装置などの中継者に追加機能として加えても良い．プロキシは，**CoAP **プロトコルの解釈のみを行い，アプリケーションデータの解釈を行わ**

ない．アプリケーションデータの解釈はクライアントとセンサーノードでのみ行い，間でデータの中継を行うプロキシではアプリケーションデータは解釈せず，各クライアントへの集約メッセージを配送する．

### 6.3.1 通信の流れ

まず，応答集約を利用しない場合の通信の流れについて記述する．図 6.2 に通信例のシーケンス図を示す．

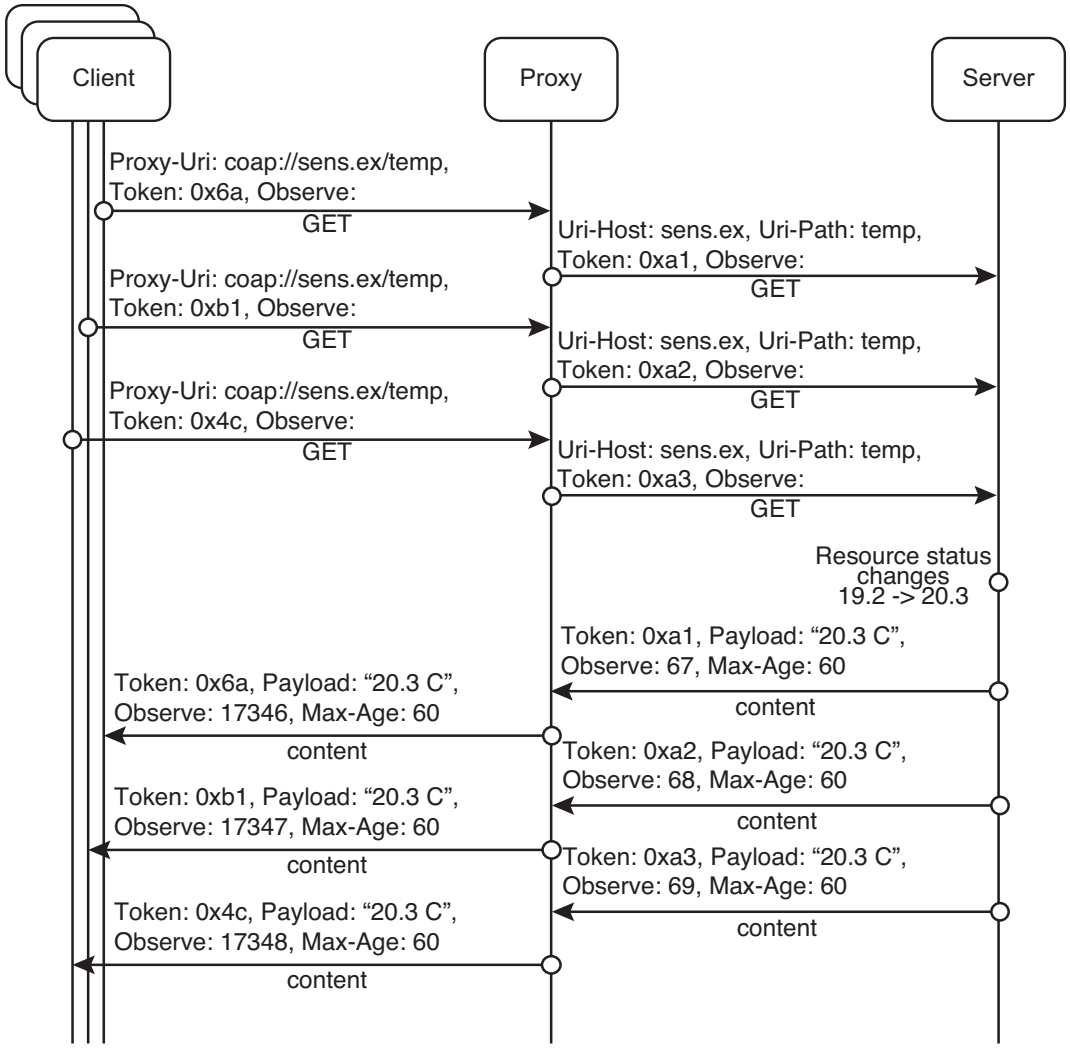


図 6.2: 応答のシーケンス

各クライアントはプロキシによって示されたリソースに対して，observe リクエスト

メッセージを送信する．このリクエストは Proxy-Uri, Token を含む GET コマンドである．

各クライアントからの observe リクエストを受信したプロキシは，それぞれのリクエストに含まれる Proxy-Uri から宛先サーバーを導き，関連する Uri-Host，そして Uri-Path とサーバーとの通信のための新たなトークンを含めたメッセージを送信する．この時プロキシは受信したメッセージに含まれるトークンと，サーバーへ送信するメッセージに含まれるトークンの関係のリストを保存する．

リソースに対する observe リクエストを受信したサーバーは，CoAP レスポンスコード 2.05 (content) の応答メッセージをプロキシに対して送信する．このメッセージには，Token, Observe, Max-Age, Payload が含まれる．ここでの Observe はサーバーによって示されるものであり，サーバーが送信したメッセージの識別に用いられる．

サーバーからの応答メッセージを受信したプロキシは，保存したトークンの関係リストから宛先クライアントを導き，リソースの変化に伴い content 応答メッセージを送信する．このメッセージは，各クライアントのリクエストメッセージに含まれていたトークンと，プロキシが示す Observe，そしてサーバーから送信されてきた Payload を含む．

プロキシからの応答メッセージを受信したクライアントは，このメッセージに含まれるトークンからリクエストとの対応関係を導き，メッセージを処理する．以上が応答集約を用いない通信の流れである．

応答集約では，サーバーとプロキシにそれぞれ以下に示す機能を追加することにより，サーバーからプロキシへの間での通信量を低減する．図 6.3 に応答集約を用いた通信の例のシーケンス図を示す．また，表 6.3 にそれぞれのメッセージに含まれる値を示す．

表 6.3: メッセージ内容

| メッセージ種別 | メッセージ内容 |
|---|---|
| クライアントからプロキシへのリクエスト | Proxy-Uri, Token |
| プロキシからサーバーへのリクエスト | Uri-Host, Uri-Path, Token |
| サーバーからプロキシへの応答 (通常) | Token, Payload |
| サーバーからプロキシへの応答 (応答集約) | Token, Token... ,Payload |
| プロキシからクライアント | Token, Payload |

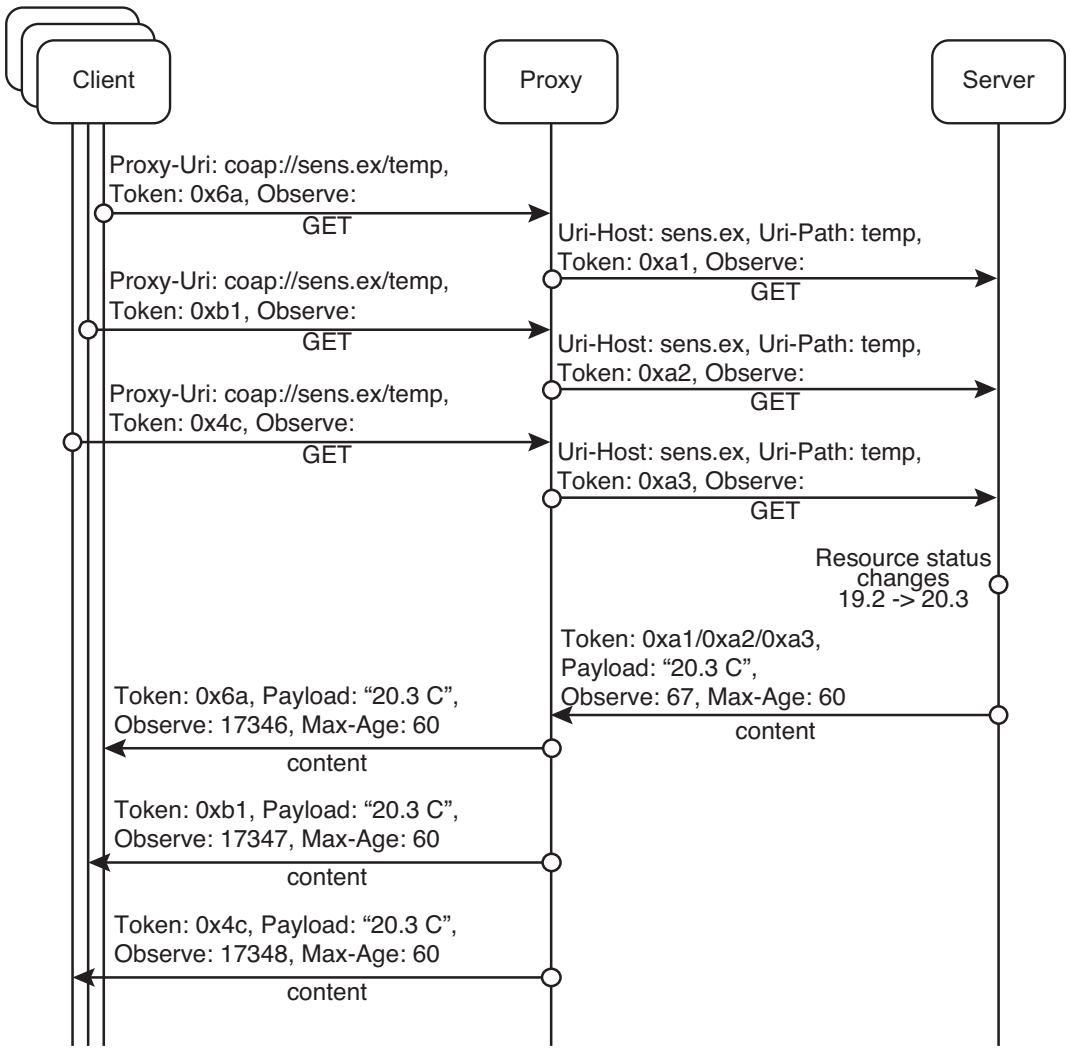


**図** 6.3: **応答集約のシーケンス**

**サーバーの機能**

**サーバーは受信したリクエストの示すリソースが応答集約可能であるかを判断し，可能な場合はそのリソースへのリクエストのトークンを保存するリストを作成する．そして，リソース状態の変化などにより応答を送信する時に作成したリストを参照し，リスト中に複数のトークンが存在する場合にその全てのトークンを含めた応答集約メッセージを送信する．**

**プロキシの機能**

**プロキシはサーバーからの複数のトークンを含む応答集約メッセージを受信すると，メッセージに含まれるトークンを元に，クライアントとプロキシの間のトーク**

ンとプロキシとサーバー間のトークンの関係リストから，宛先クライアントを導き，それぞれのクライアントに対して応答メッセージを送信する．

プロキシは Proxy-Uri から Uri-Host と Uri-Path を導くため，これを利用してリクエストの段階で集約が可能に見える．しかし，各 uri が応答集約が可能であるかは uri のみでは判断できず，プロキシにおいて集約はできない．プロキシにおいて集約を行う場合は，プロキシは各サーバーのリソースが集約可能であるかの情報を事前に知る必要があり，この情報の交換機能を全てのプロキシとサーバーに追加する必要がある．提案する応答集約では，プロキシは既に備わっているトークンの対応関係のみを利用しているため，機能の追加を必要としない．

## 6.4 本章のまとめ

本章では，多様なアクセス形式にまたがる End-to-End 通信を行う IoT システムで利用できる応答集約手法について述べた．また，提案手法の RACOW システム上と CoAP 上での実現手法について述べた．

次章では提案した応答集約の効果の評価について述べる。

# 第 7 章

# 提案手法の評価

本章では，第 6 章で述べた応答集約手法の End-to-End 通信を行う IoT システムにおける有効性評価について述べる．

## 7.1 評価方針

提案する応答集約機能を適用した場合と適用しない場合，それぞれにおけるワイヤレスセンサーネットワークの通信状況を比較することで，提案手法の評価を行った．評価実験では実機への機能実装を行い，実際のワイヤレス環境下で測定を行った．

## 7.2 実験環境

評価実験には第 2.3.1 節と同様の機材を用いた．クライアントはゲートウェイを経由してセンサーデバイスにリクエストを送信する．プロキシの機能はゲートウェイ装置に追加し，各センサーデバイスに対するリクエストはゲートウェイ装置内部のリストに保存される．リクエストは定期間隔で 50 byte のセンサーデータを送信する観測リクエストとし，センサーデータの送信間隔は 1 秒に 1 回とした．また，クライアントは応答データを内部に蓄積する．以上に述べた環境において，10 秒置きに全てのセンサーデバイスに対して新しいリクエストを 20 回まで出し，各リクエストに対応する Acknowledge をクライアントに受信したかを測定した．ワイヤレス区間は SNA を用いて観測した．センサーデバイスの数を 1 台の場合と 4 台の場合の 2 通りの実験を行った．

### 7.2.1 実験結果

まず，使用センサーデバイス数が 1 の場合の実験結果を図 7.1 に示す．図から応答集約を行わない場合には，リクエスト数が 5 以下の場合にはリクエストが受理される．これを超えたリクエストがある場合には受理の失敗が見られ，最終的に受理されたリクエストの数は 8 であった．SNA の測定結果から，センサーデバイスがそれぞれのリクエストに対して，センサーデータを送信するのにかかる時間は 190 msec 程度だとわかった．リクエスト数が 5 以上になると，既存のリクエストに対する応答処理にかかる時間がセンサーデータの送信間隔である 1 秒の大部分を占めるため，新規のリクエスト受理が頻繁に失敗していると判断できる．

一方，応答集約を用いた場合はリクエスト数が 18 以下の場合には全て受理されるが，それを超えたリクエストについては受理が失敗した．20 回目のリクエストが送信されてから 1 分後に，SNA を用いてワイヤレス区間におけるパケットの構成を見ると，パケットに含まれる eID は 18 回目のリクエストまでの値であり，19 回目，20 回目のリクエストについてはセンサーデバイスによって破棄されているとみなせる．これは，センサーデバイスが持つ eID のリストのデータ量が利用可能なバッファ領域を超過したためと考えられる．

この実験におけるセンサーデータ送信の通信量の変化を図 7.2 に示す．応答集約を用いない場合では，新規にリクエストが受理されるごとに送信するパケットが 1 つずつ増えるため，通信量は 808 bit ずつ増加する．一方，応答集約を用いる場合の通信量増加はパケットへの新規追加 eID の値のみのため，3 byte ずつの増加となる．応答集約適用時には，追加 eID 値の増加によって，ひとつのパケットの持てるペイロードのサイズをメッセージのデータ量が超過する場合があり，その際にはメッセージを複数のパケットに分割して送信する．この実験ではリクエスト数が 11 以上の場合がこれにあたり，パケットの分割により通信量は 384 bit 増加する．

次に，使用センサーデバイス数が 4 の場合の実験結果を図 7.3 に示す．図から応答集約を行わない場合には，リクエスト数が 4 以下の場合にはリクエストは受理されるが，これを超えたリクエストがあるとクライアントは受理の失敗をする．

一方，応答集約を用いた場合は 14 回目のリクエストまでは成功しているが，15 回目と 17 回目以降のリクエストについては失敗している．20 回のリクエスト送信から 1 分後に，SNA を用いてワイヤレス区間におけるパケットの構成を見ると，センサーデバイス数 1 の場合と同様に 18 回目までのリクエストに対応する eID が全て含まれていた．こ

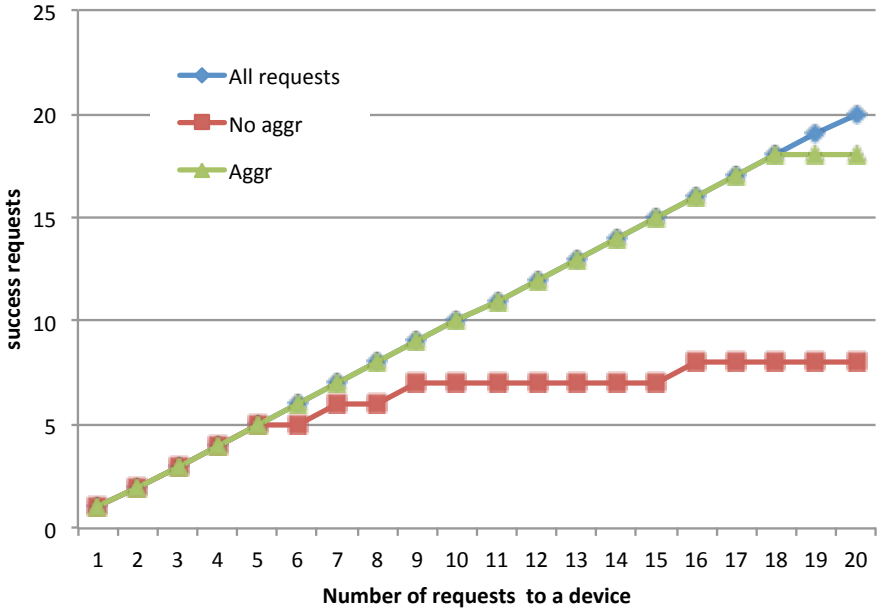


**図 7.1: ノード数 1 のリクエスト受理数**

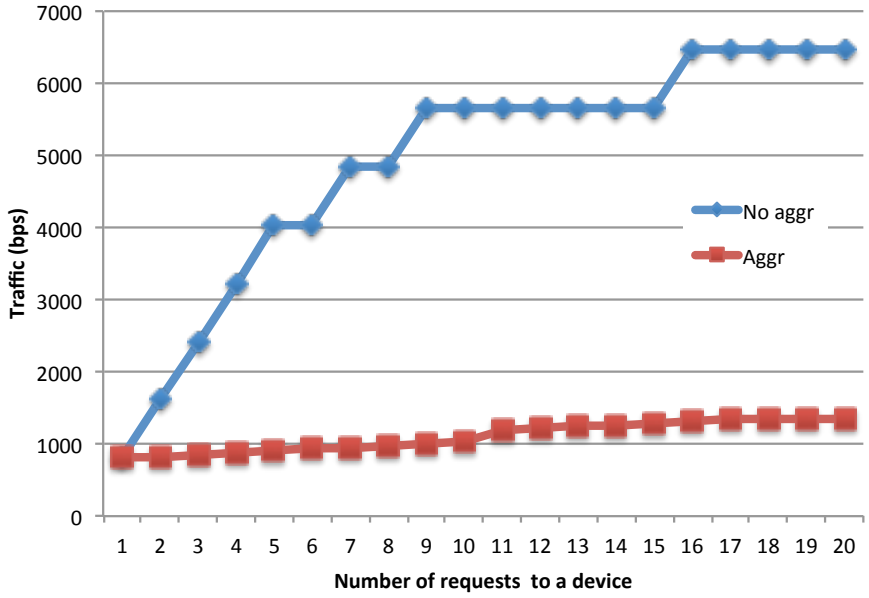


**図 7.2: ノード数 1 の通信量**

の受理失敗はセンサーデバイスからの，受理完了メッセージがゲートウェイ装置において失われいるために発生していると判断できる．本実験では集約メッセージの分配はゲートウェイ装置によって行われており，リクエスト数の増加に伴いゲートウェイが負荷は大きくなったことが原因と判断できる．

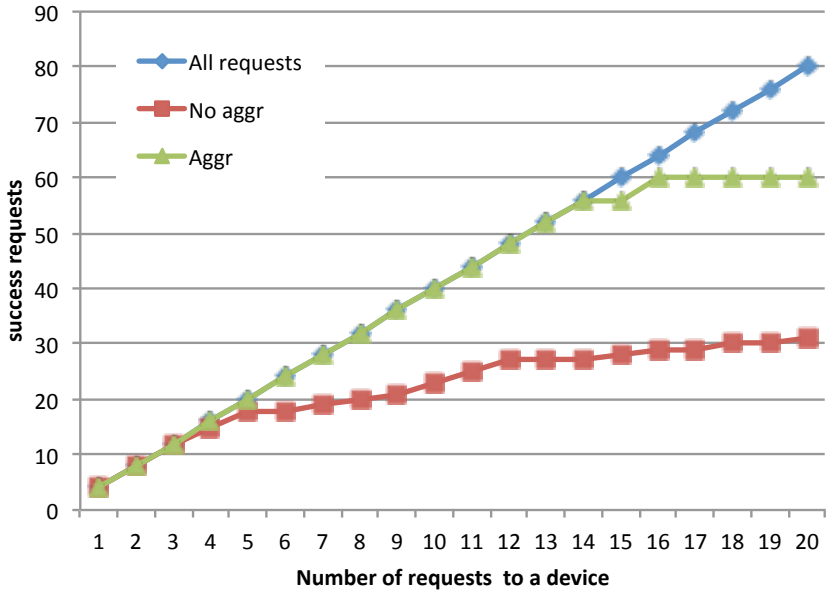


図 7.3: ノード数 4 のリクエスト受理数

## 7.3 本章のまとめ

本章では，第 6 章で提案した応答集約手法の有効性評価について述べた．実機を用いた実験により応答集約を用いることで，同じリソースデータを 1 秒間隔で送信するリクエストについて，1 つのセンサーデバイスとの通信量を処理リクエスト数 8 の場合において，6464 bps から 976 bps へと低減できることを明らかにした．また，リクエストの受理可能数を 1 つのセンサーデバイスでは 8 から 18 へと改善でき，4 つのセンサーデバイスでは 31 から 60 へと改善できることを明らかにした．

# 第 8 章

# おわりに

## 8.1 本論文の結論

本論文では，複数のアクセスネットワークにまたがって End-to-End 通信を行う Internet of Things システムにおいて，センサーデバイスが重複してデータを送信することで輻輳が発生する問題に着目した．また，HTTP/IP および CoAP/ZigBee を用いてデータを収集する環境を構築し，情報システムにおける性能ボトルネックを実験的に調査し，その結果，定期応答メッセージが輻輳の原因となることを示した．その解決案として，ゲートウェイ装置に等棟集約機能を加えたプロキシ機能を追加し，がアプリケーションデータの内容を解釈すること無しに，センサーノードのがセンサーデータの応答集約を可能とする手法を提案した．提案手法は ZigBee を用いた実装により，同じリソース情報を 1 秒間隔で送信する応答について，1 つのセンサーデバイスが応答数が 8 の場合において，通信量を 6464 bps から 976 bps へと低減でき，4 つのセンサーデバイスが連携できるクライアントの数を合計 31 から 60 へと改善できることを明らかにした．

# 参考文献


[1] Z.Shelby, K.Hartke, C.Bormann, and B.Frank. Constrained application protocol (coap) draft-ietf-core-coap-13. internet draft, December 2012.

[2] ZigBee Alliance. Smart energy profile 2.0. http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Version20Documents.aspx.

[3] Racow project @ONLINE 2013/1/9 **アクセス**. http://www.racow.net/.

[4] Drew Gislason. Zigbee technical overview, July 2008. http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=13710.

[5] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007. Updated by RFCs 6282, 6775.

[6] K.Hartke. Observing resources in coap draft-ietf-core-observe-07. internet draft, October 2012.

[7] Jrg Widmer Elena Fasolo, Michele Rossi and Michele Zorz. Data aggregation techniques in sensor networks: A survey. *IEEE Wireless Communications Magazine*, pages 70–87, 2007.

[8] K. Obraczka I. Solis. The impact of timing in data aggregation for sensor networks. *IEEE ICC*, pages 3640 – 3645 Vol.6, 2004.

[9] C. Xiaojun F. Hu and C.May. Optimized scheduling for data aggregation in wireless sensor networks. *IEEE ITCC*, April 2005.

[10] S. Madden et al. Tag: a tiny aggregation service for ad hoc sensor networks. *OSDI*, Dec 2002.

[11] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Sys.*, pages 924 – 35 Vol.13, Sept 2002.

[12] Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, Ros Labrinidis, and Panos K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks, 2004.

[13] Erdal Cayirci. Data aggregation and dilution by modulus addressing in wireless sensor networks. *IEEE Communications Letters*, pages 355–357, 2003.

[14] Tian He, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Aida: Adaptive application independent data aggregation in wireless sensor networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 2004.

# 謝辞

本論文の作成にあたり，御助言をいただきました慶應義塾大学環境情報学部教授村井純博士，同学部教授中村修博士，同学部准教授楠本博之博士，同学部専任講師 Rodney D. Van Meter III 博士，同学部准教授植原啓介博士，同学部教授武田圭史博士に感謝します．

研究に関する指導を多くいただきました，研究環境情報学部准教授 三次仁博士に深く，深く，深く，感謝いたします．

五年という長い歳月を過ごした Auto-ID Lab. Japan という場所に深く感謝いたします．

Auto-ID Lab. Japan での研究活動で様々な助言をいただきました稲葉達也博士，羽田久一博士，鈴木茂哉博士，中根雅文氏に深く感謝いたします．Auto-ID Lab. Japan での研究生活を共に過ごしたパナソニックシステムネットワークス株式会社 徳増理氏，電気通信大学大学院情報理工学研究科総合情報学専攻助教 川喜田佑介博士，株式会社日放電子 白石雅彦氏，ホロシェフ ロマン氏，株式会社ブロードバンドタワー 青木伸行氏，神谷尚保氏，電機工業株式会社 松本伸史氏，佐藤泰介氏，佐藤龍氏，江村桂吾氏，杉本健一氏，山口修平氏，金仙麗氏，鈴木詩織氏，田村哲朗氏，富田千智氏，廣石達也氏，山田真弘氏，能島良和氏，佐藤友紀氏，横石雄大氏，Doan Hoai Nam 氏，小澤みゆき氏，吉田守氏，三樹良亮氏，五十嵐祐貴氏清水真有氏，小薗宏樹氏，城風智氏，百石順一朗氏，鈴木駿氏，小畠大平氏，渡辺至都氏に感謝いたします．

修士論文執筆を共にした，宮崎圭太氏，三部剛義氏，佐藤弘崇氏に感謝いたします．お疲れ様でした．

執筆活動と常に共にあった楽曲を制作した Steve Reich 氏に感謝いたします．

そして、徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田合同研究プロジェクトの諸氏皆々様に感謝致します。

以上を持って謝辞とさせていただきます．ありがとうございました．