# データ量と通信品質を考慮した IoT 向け優先度制御手法の実装と評価

橘拓馬 [1] 古都哲生 [2] 峰野博史 [3]

**概要**：センサや通信モジュールの小型化と，LTE 等の様々なモバイルネットワークの出現によって，Internet of Things (IoT)への期待が高まりつつある．ネットワークに接続されるデバイス（以下，IoT デバイス）は多岐に渡り，やり取りされるデータも多種多様（テキスト，画像，音声，センサ，位置等）かつ膨大な量になると想定されている．それに伴い多数の IoT 向け通信手法や通信プロトコルの検討が行われているが，様々な特性を持つデータや，多種多様な品質のモバイルネットワークで構築される IoT システムを意識した IoT 向け通信制御手法については十分な検討が行われているとは言えない．本研究では，通信品質が動的に変動するモバイルネットワークで，アプリケーションの要求する QoS を満たす IoT 向け優先度制御手法を提案する．提案手法では制御サーバの役割を果たす Broker サーバが優先度を用いて IoT デバイスのデータ送信タイミングを指示する．送信優先度は，アプリケーションサーバ（AP サーバ）上のアプリケーションの QoS 要求と IoT デバイスが生成するデータの特性（遅延耐性など）に応じて決定される．詳細設計した IoT 向け優先度制御手法を実装し，動的に通信品質が変動するモバイルネットワーク上で有効性を評価したところ，高優先度データの送信率を最大 75.0%上昇させられ，平均遅延時間は最大 10.7%削減できた．

# Implementation and Evaluation of Priority Control Mechanism for Heterogeneous IoT Environment

TAKUMA TACHIBANA[1] TETSUO FURUICHI[2] HIROSHI MINENO[3]

## 1. はじめに

センサや通信モジュールの小型化と，BLE や LTE，Wi-Fi といった様々なモバイルネットワークの出現によって，Internet of Things（IoT）[1]への期待が高まりつつある[2,3]．IoT の発展に伴い多数のオブジェクトやデバイス（以降，IoT デバイス）がモバイルネットワークを介して接続されると見込まれており[4]，2014 年から 2019 年の 5 年間で IoT デバイス数は約 7 倍に，IoT デバイス一台当たりの通信量は約 5 倍に増加すると予想されている．今後も様々な IoT デバイスが出現すると考えられ，温度や湿度，位置情報といったセンサデータだけでなく，画像や音声といった多種多様なデータ通信もが想定される．

一方，他通信事業者から設備を借りて独自通信サービスを展開する Mobile Virtual Network Operator （MVNO）が増加してきている．MVNO の通信サービスは，これまでスマートフォン向けを主軸にしてきたが，近年は IoT 向けサービス[5]も登場し，料金低廉化だけでなく多種多様なサービス形態が出現しつつある．これら IoT 向けサービスは，契約内容によっては帯域保証されておらず，大量又は大容量データ通信に不適であり，送信データの大幅な遅延や欠落の生じる可能性がある．そのため，**図 1** に示すような多種多様かつ多数のセンサやカメラを利用したリモートモニタリング IoT システム[3]での使用を考えると，システムの可用性や信頼性低下が懸念される．

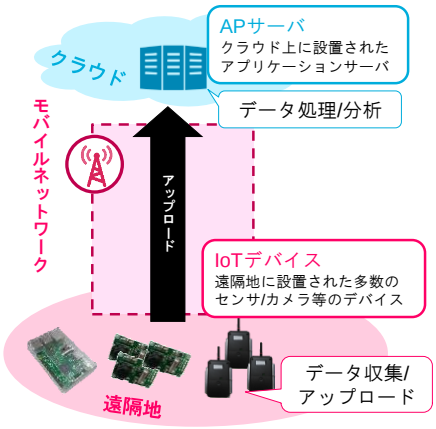


**図 1** リモートモニタリング IoT システムの例

本研究では，多種多様かつ多数のセンサやカメラの送信するデータの持つ異なる特性（遅延耐性）に着目し，アプリケーションの要求する Quality of Service（QoS）に応じて送信データ量を制御する IoT 向け優先度制御通信手法を提案する．

以下，本論文の構成を述べる．第 2 章で関連研究について整理し，第 3 章でデータ量と通信品質を考慮した IoT 向け優先度通信手法を提案する．第 4 章で実装と評価結果，第 5 章でまとめと今後の課題を述べる．

1 静岡大学大学院総合科学技術研究科
Graduate School of Integrated Science and Technology, Shizuoka University
2 株式会社メガチップス ソフトウェア研究所
Software R&D Laboratory, MegaChips Corporation
3 静岡大学学術院情報学領域 / JST さきがけ
College of Informatics, Shizuoka University / JST, PRESTO

## 2. 関連研究

近年，様々な IoT 向け通信プロトコル・通信手法が提案されている[6-10]．MQTT[11]は，小容量・高頻度・多数のクライアントのトラフィックを持つ IoT システムの特徴に着目した IoT 向け通信プロトコルである．MQTT のプロトコルヘッダは最小 2 バイトと軽量であるため，センサ値など小容量データを送信する際に通信量を削減する方法として有効である．しかし，画像や音声などの大容量データを送信する場合，データ全体に対するヘッダ部の通信量の割合が低いため，ヘッダ軽量化の効果はそれほど大きくないという特徴ある．また，MQTT は不安定なモバイルネットワークを想定した QoS 制御機能があるが，この機能はデータの重要度に応じて再送制御を行うことでデータの到達を保証する機能であり，データの特性（遅延耐性）まで考慮した優先度制御までは行っていない．農業や鳥獣観測のような IoT アプリケーションでは，周期的なデータ収集が望まれるが，データ量と通信品質に応じて 30 分毎のデータ収集のみ確実にできれば，その間の 5 分毎のデータ収集は多少欠損あっても問題ないという場合もある．IoT アプリケーション開発者が，このような QoS を満たすようにソフトウェアを独自実装しなくても，優先度を設定しておけばデータ量や通信品質に応じて自動的に優先度制御してくれるミドルウェアがあると便利である．

一方，モバイルネットワークにおけるデータトラフィックの時間的局所性や空間的局所性を平滑化するために，遅延耐性のあるデータに対して時間的・空間的・通信路的に負荷分散させるモバイルデータオフローディングプロトコルとして，MDOP が提案されている[12]．MDOP は，様々なデータが持つ異なる遅延耐性に着目した通信プロトコルで，遅延耐性の高いデータを一時的に蓄積し，基地局の負荷状況やデータの遅延耐性を考慮して，送信タイミングを遅らせて送信レートを制御する．現在，研究開発が進められているが，多種多様な IoT デバイスのデータの特性までは未検討で，データ量や通信品質によっては送信レート制御だけでなく優先度制御の適用も効果があると考える．
また，CoAP[13]は，Hypertext Transport Protocol（HTTP）で一般的となった URI や HTTP レスポンスコードなど，Web 技術の特徴を踏襲して IoT 向けに軽量化・簡易化した通信プロトコルである．CoAP では，HTTP ヘッダをバイナリ化してヘッダ軽量化を実現しているだけでなく，トランスポート層プロトコルとして軽量な UDP を用いることでオーバヘッドを削減している．ただし，UDP を用いたオーバヘッド削減は，通信の確実性とトレードオフの関係にあり，正確かつ確実にデータ収集するためには再送制御や QoS 制御を IoT アプリケーション層で実装する必要がある．

以上のように，IoT システム向け通信手法の研究開発は多数行われているが，今後の IoT システムでは「小容量・高頻度・端末数増大」を考慮した通信手法[14]に加え，様々な特性を持つデータと多種多様な品質のモバイルネットワークを考慮した IoT 向け通信手法が重要になると考える．

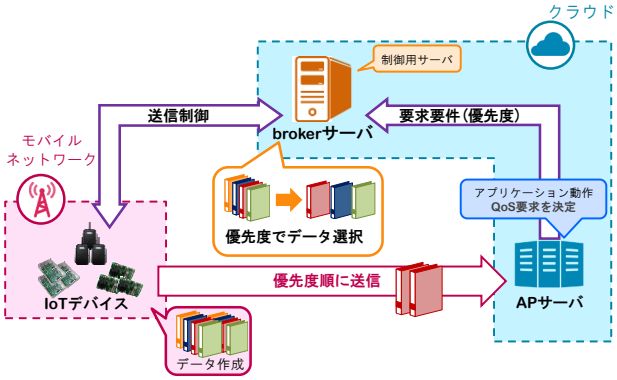


**図 2** IoT 向け優先度制御手法の全体像

## 3. 提案手法

### 3.1 データ量と通信品質を考慮した IoT 向け優先度制御

サービス品質が動的に大きく変動しがちな IoT 向け通信サービスに対し，多種多様なデータの持つ遅延耐性や優先度に着目し，IoT アプリケーションの要求に応じて適切に送信データの優先度制御や送信データ量の削減を行う IoT 向け優先度制御手法を提案する．提案手法では，設定に基づいて全ての送信データに対し優先度制御を行うことで，ネットワーク帯域の変動に追従可能な通信量を制御する．これにより，動的に品質変動するモバイルネットワークであっても，IoT アプリケーションの要求する QoS を満たす優先度制御を実現する．

**図 2** に提案する IoT 向け優先度制御手法の全体像を示す．IoT 向け優先度制御手法は，モバイルネットワークを介してインターネットに接続される IoT デバイス，IoT デバイスからの受信データを処理するアプリケーションサーバ（以降，AP サーバ），優先度制御を行う Broker サーバの大きく 3 要素によって構成する．ここで，AP サーバと Broker サーバは機能的な分割であり，同一物理マシン上に配置しても構わない．Broker サーバと AP サーバと明示的に分割することで，コントロールプレーンとデータプレーンを分けて説明しやすいだけでなく，複数 AP サーバ導入といった拡張にも容易に対応できる．

**図 3** に IoT 向け優先度制御手法の動作フローを示す．IoT 向け優先度制御手法は，登録フェーズ，優先度通信フェーズ，解除フェーズといった大きく 3 つのフェーズから構成される．登録フェーズでは，IoT デバイスが自身の IoT デバイス情報を Broker サーバへ登録するだけでなく，AP サーバは自アプリケーションの QoS 要求を Broker サーバへ登録する．Broker サーバは優先度テーブルを作成するとともに，IoT デバイスへ AP サーバ情報を伝える．これによ

り，IoT デバイス，AP サーバ，Broker サーバ間相互で TCP コネクションを確立する．優先度通信フェーズでは，Broker サーバは IoT デバイスの生成したデータ（以降，コンテンツデータ）に優先度を付与し管理する．Broker サーバは IoT デバイス全体の送信レートを確認し，コンテンツデータに送信レートの一部を割り当てるか，送信タイミングを遅延させて優先度制御を行う．最後に解除フェーズでは，アプリケーションからの指示により各種コネクションの解除と優先度キュー，各種バッファの初期化を行う．

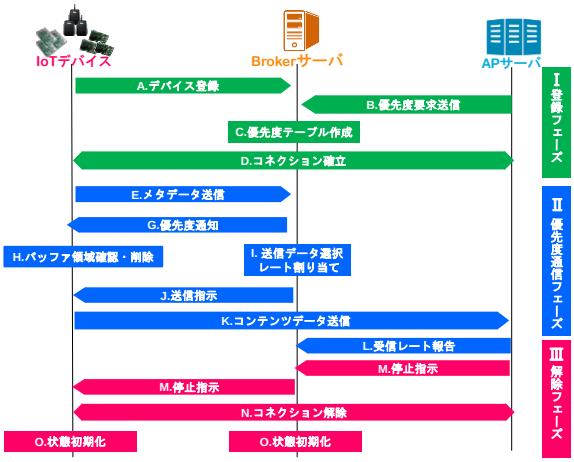


**図 3** IoT 向け優先度制御手法の動作フロー

### 3.2 想定する IoT アプリケーション

本稿で提案する IoT 向け優先度制御手法は，モバイルネットワークと IoT デバイスを用いて遠隔地の情報収集を行うリモートモニタリング IoT システムへの適用を想定している．例えば，農業支援システム[15,16]などでは，現場の撮影画像も収集でき様子をリモートで把握できることが望ましい．理想的には，単なる状況把握だけでなく，詳細な様子を柔軟にリモートモニタリングできることが望ましいが，カメラなどの IoT デバイスを多数用いると，大容量データの送信によってモバイルネットワークの許容量を超えてしまい，温度や湿度といったデータサイズの小さい環境データの送信に不都合を生じることがある．動的に回線品質の変動するモバイルネットワークに対し，様々なデータ送信の適切な粒度を事前に設定しておくことは困難なため，データ量や通信品質に応じて自動的に優先度制御されることが望ましい．提案する IoT 向け優先度制御手法を実現する機能をライブラリとして実装し，このライブラリを用いた IoT アプリケーションを開発すれば，IoT アプリケーション固有の遅延耐性や優先度に対する QoS に基づいて必要最低限のデータ通信のみ保証するベストエフォート型のリモートモニタリング IoT システムを実現できる．

### 3.3 登録フェーズ

まず，以降で用いる IoT 向け優先度制御手法の用語一覧を**表 1** に示す．本研究では，各構成要素の識別子として DeviceID が付与済みであることを想定しているが，説明を簡単にするため Broker サーバに 0x00，AP サーバに 0xff，IoT デバイスに 0x01~0xfe の間の固定値を設定済みとする．また，IoT デバイスが送信するコンテンツデータのプロファイルであるメタデータ，IoT 向け優先度制御手法で通信するすべてのデータをメッセージと呼ぶこととする．Broker サーバでは，IoT デバイスが送信するデータの優先度決定ルールを記述した「優先度決定テーブル」と，IoT デバイスと AP サーバとの接続状況を管理する「管理テーブル」を保持し，これら 2 つのテーブル内の情報と IoT デバイス，Broker サーバからの情報を用いて優先度制御を行う．

登録フェーズは，提案する IoT 向け優先度制御手法で最初に行われるフェーズである．**図 4** に登録フェーズの詳細を示す．優先度通信フェーズに必要な情報として，IoT デバイスは Broker サーバへ DeviceID と送信するデータの種類を示す DataType を登録する（図 4 の A）．一方，AP サーバは IoT アプリケーションの QoS 要求を Broker サーバへ送信し，優先度決定テーブルが更新される（図 4 の B）．その後， Broker サーバは，IoT デバイス，Broker サーバ，AP サーバ間に相互コネクションを確立する．以上によって，優先度通信フェーズで必要となる各デバイスやサーバの接続状態と IoT アプリケーションの QoS 要求を Broker サーバが集約し，IoT 向け優先度制御手法を実現する．

**表 1** IoT 向け優先度制御手法の用語一覧

| 用語 | 説明 |
|---|---|
| DeviceID | IoT デバイスの識別子 |
| DataType | コンテンツデータの種類(ユーザ指定) |
| コンテンツデータ | IoT デバイスが送信するデータ |
| メタデータ | コンテンツデータのプロファイル．<br>コンテンツデータと 1 対 1 で紐づく |
| メッセージ | 優先度制御手法で通信するすべてのデータ |
| 優先度決定テーブル | データの優先度を決定するルールを<br>記述したテーブル |
| 管理テーブル | IoT デバイス・AP サーバの接続状況を<br>管理するテーブル |

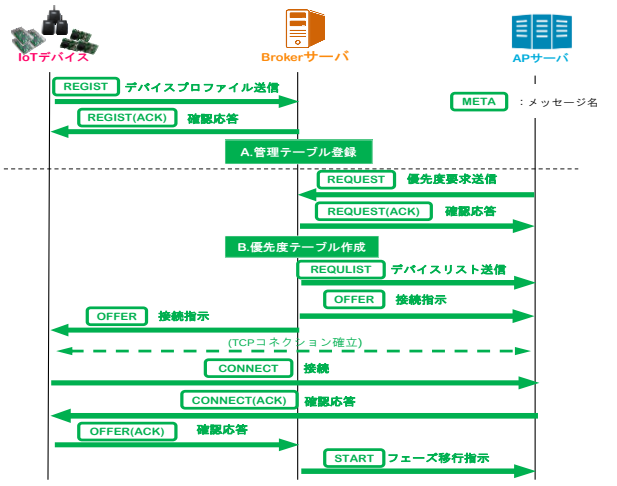


**図 4** 登録フェーズの詳細

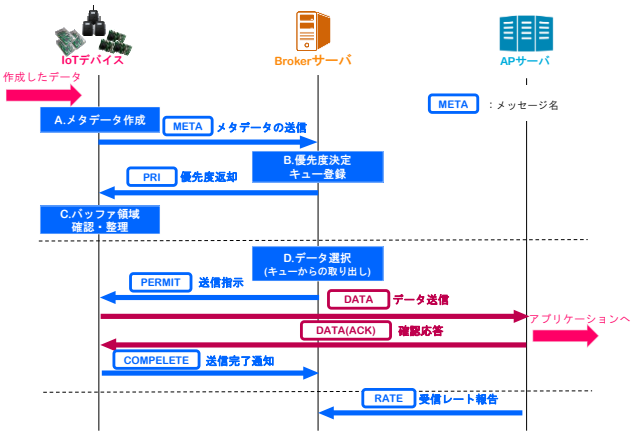


**図 5** 優先度通信フェーズの詳細

### 3.4 優先度通信フェーズ

優先度通信フェーズは, IoT デバイスが生成したコンテンツデータへ優先度を設定し，優先度通信を行うフェーズであり，本提案手法の主となるフェーズである．**図 5** に優先度通信フェーズの詳細を示す．優先度通信フェーズでは，コンテンツデータを送信する際に優先度を付与し，優先度の高いデータから送信する．コンテンツデータに付与する優先度は Broker サーバの指示で制御され，Broker サーバは接続する全 IoT デバイスが送信希望しているコンテンツデータと IoT デバイス全体の送信レートを確認しながらコンテンツデータ毎の送信レート調整や，送信タイミングを遅延させて優先度制御を行う．この際，IoT デバイスは送信前のコンテンツデータを一時的にバッファリングする．バッファ領域が不足した場合は低優先度のコンテンツデータから送信を行わずに削除する．これにより動的に変動する通信品質に合わせた送信量の制御を実現する．

以下に動作フローの詳細を述べる．IoT デバイスは自身の生成したコンテンツデータを送信する前に，コンテンツデータに DataID を付与し，メタデータを作成する（図 5 の A，詳細は後述）．IoT デバイスは作成したメタデータを META メッセージとして Broker サーバに送信する．コンテンツデータはこの時点ではまだ送信しない．Broker サーバは META メッセージを受信すると，優先度決定テーブルとメッセージ内のメタデータを照合して優先度を決定する（図 5 の B，詳細は後述）．優先度を決定すると Broker サーバは，IoT デバイスへ優先度を PRI メッセージで返し，続けて自身の優先度キューにメタデータ，DataID，優先度を記録する．一方, IoT デバイスは PRI メッセージ受信後，コンテンツデータ，DataID，優先度を自身のバッファに保存する．このとき IoT デバイスのバッファ内の容量が閾値を超えた場合，優先度の低いデータから削除する．バッファに保存したコンテンツデータなどは，後述する Broker サーバからの送信許可を表す PERMIT メッセージを受信するまで待機する．また，IoT デバイスからの META メッセージを受信した Broker サーバは，優先度キューにメタデータが記録されている場合，優先度キュー内で最も高い優先度を持つメタデータをキューから取り出し，現在の通信レートに応じた送信レートを割り当てる（図 5 の C，詳細は後述）．送信レートを割り当てたら，Broker サーバは PERMIT メッセージで IoT デバイスにコンテンツデータの送信を指示する. PERMIT メッセージを受け取った IoT デバイスは，メッセージ内の DataID で指示されたコンテンツデータを AP サーバへ DATA メッセージを用いて送信する．AP サーバは IoT デバイスからデータを受信しつつ，受信データの大きさと受信時間から受信レートを計算し，周期的に Broker サーバに通信レートとして報告する．

優先度通信フェーズ動作中は上記動作を繰り返すことで，動的変動するデータ量と通信品質を考慮して IoT アプリケーションの QoS 要求に応じた優先度通信を実現する．

### 3.5 解除フェーズ

解除フェーズは，提案する IoT 向け優先度制御手法の終了する際の最後のフェーズであり，IoT アプリケーションの指示を受けて IoT デバイスと Broker サーバの状態を初期化する．**図 6** に解除フェーズの詳細を示す．IoT アプリケーションからの指示を受け，Broker サーバは IoT デバイスと AP サーバ間のコネクションの解除を指示するとともに，自身の該当する優先度テーブルの初期化を行う（図 6 の G）．一方，解除指示を受信した IoT デバイスも，該当する優先度キュー，バッファの初期化を行う．最後に AP サーバと Broker サーバ間の TCP コネクションを解除する．

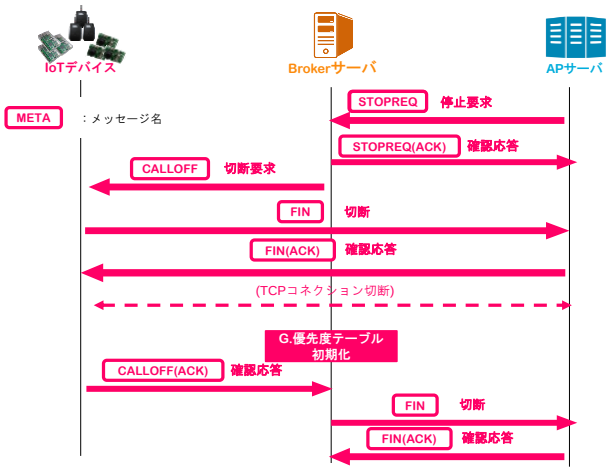


**図 6** 解除フェーズの詳細

### 3.6 優先度決定の詳細

Broker サーバで調整するコンテンツデータの優先度決定方法の詳細を述べる. 各コンテンツデータの優先度は, 1 を最も高い優先度として 15 段階で表すものとする．この優先度の決定要素は，「送信デバイス」「送信データの種類」「送信時刻」の 3 つとした. これら 3 要素を用いることで，「いつ」「どのデバイスが」「どのようなデータを」送信し

ようとしているのか判断でき，優先度制御を実現する．この優先度決定には，IoT デバイスから送信されたメタデータと Broker サーバ上の優先度決定テーブルを用いる．

メタデータは，コンテンツデータと 1 対 1 で紐づけられるコンテンツデータの性質を表した情報であり，IoT デバイスがコンテンツデータを作成した時に作成される．メタデータの構成を**表 2** に示す．メタデータはデータを識別する DataID と DeviceID のほか, データ容量を示す DataSize, データの種類を示す DataType を含む．ここで，IoT デバイス間の正確な時刻同期は困難なため，IoT デバイスの送信するメタデータには時刻情報を含めず，Broker サーバにメタデータが到着した時刻をコンテンツデータの生成時刻とみなして優先度決定を行う．メタデータの DataType を除いた大きさは 5Byte とし，仮に DataType を 3 文字で表現したとしても 9Byte と非常に小さいため，通信帯域を圧迫せずに Broker サーバへコンテンツデータの情報を送信可能である．

Broker サーバが保持する優先度決定テーブルは，メタデータの要素からコンテンツデータの優先度を決定するためのテーブルである．優先度決定テーブルの構成を**表 3** に示す．表の行番号を表す ColumnID のほか，送信元デバイスを示す DeviceID，データの種類を示す DataType，送信間隔を示す DataInterval がある．

優先度決定テーブルの例を**表 4** に示す．優先度決定テーブルは，DeviceID，DataType，DataInterval のいずれかの条件が異なっていれば異なるカラムとして扱う．条件ごとに異なるカラムとして扱うことで「デバイス 1 とデバイス 2 で同じ種類のデータを送信するが，デバイス 1 のデータをより重視する」ことや「同一のデバイスから送られる異なる種類のデータを異なる優先度で扱う」，さらに「同じデバイスからのデータでも 10 分に 1 回は優先度を高く，それより短い間隔の場合は優先度を低く設定する」など，通信のセッション毎の制御だけでなく，柔軟かつ詳細な IoT アプリケーション向け優先度制御を実現する．

**表 2** メタデータの構成

| フィールド名 | 説明 | 形式 |
|---|---|---|
| DataID | データ識別子 | 16bit 符号無整数 |
| DeviceID | 送信デバイスの識別子 | 8bit 符号無整数 |
| DataSize | データの容量 | 16bit 浮動小数点 |
| DataType | データ種類 | 文字列 |

**表 3** 優先度決定テーブルの構成

| 種別 | フィールド名 | 説明 | 形式 |
|---|---|---|---|
| 条件 | ColumnID | 表の行番号 | 8bit 符号無整数 |
| | DeviceID | 送信元デバイスの識別子 | 8bit 符号無整数 |
| | DataType | データ形式 | 文字列 |
| | DataInterval | 送信間隔 | UNIXTIME |
| 出力 | Priority | 優先度(1 を最高とする) | 1～15 の整数 |

**表 4** 優先度決定テーブルの例

| ColumnID (C) | DeviceID (D) | DataType (T) | DataInterval (I) | Priority |
|---|---|---|---|---|
| 1 | 1 | jpg | 10 分 | 2 |
| 2 | 1 | jpg | 1 分 | 6 |
| 3 | 2 | txt | 10 秒 | 1 |
| 4 | 2 | jpg | 10 分 | 6 |

※フィールド名の括弧内は図 7 で用いる略称を示す.

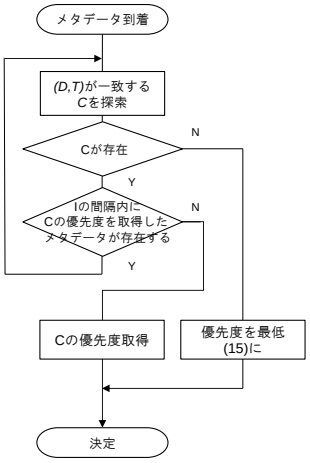


**図 7** 優先度決定アルゴリズム

優先度決定テーブルを用いた優先度決定アルゴリズムを**図 7** に示す．IoT デバイスから送信される META メッセージでコンテンツデータの条件を受け取った Broker サーバは，優先度決定テーブルを ColumnID の上位から順番に META データの条件と一致するか検索する．検索を行って最初に一致したカラムに記述された優先度を META メッセージに対応するコンテンツデータの優先度とする．最後まで一致するデータが存在しない場合は最低優先度の 15 を付与する．

### 3.7 データ選択，送信レート計算の詳細

Broker サーバは優先度制御を行うため，Broker サーバの優先度キューに存在するすべてのデータの中から優先度の高いデータを選択し，IoT デバイスに送信許可を与える．選択するデータは最も高い優先度を付与されたデータである．最も高い優先度を付与されたデータが複数存在する場合，現在の送信レートを参照しながら以下のルールに従って送信許可を与える．選択されたメタデータは，データ容量に基づいて S,M,L の 3 つにクラス分けする．クラス一覧を**表 5** に示す．クラス毎に異なる式を用いてコンテンツデータを送信時の送信レートを算出する．割当て可能な全帯域幅を $BW$，クラス $x$ のデータ数を $n_x$ としたとき，クラス

**表 5** クラス一覧

| クラス | 想定データ例 |
|---|---|
| S | 軽量なテキストデータ |
| M | 軽量な画像 |
| L | 大容量画像 |

S,M,L の各送信レート $W_s$，$W_m$，$W_l$ を，(式 1)，(式 2)，(式 3) で求める．ただし，$W_s$，$W_m$，$W_l$，はそれぞれ下限と上限を設け，$W_s$，$W_m$，$W_l$，が下限未満の場合，下限を上回るように選択するデータ数を制限する．クラス別に送信レートを調整することで，大容量データによる通信路の占有を防ぎ，同一優先度を持つデータの公平性を確保する．

$$W_S = \frac{BW}{n_s} \quad \text{(式 1)}$$

$$W_m = \frac{BW - \sum_{n_s} W_S}{n_m} \quad \text{(式 2)}$$

$$W_l = \frac{BW - \left(\sum_{n_s} W_S + \sum_{n_m} W_m\right)}{n_l} \quad \text{(式 3)}$$

## 4. 実装・評価実験

### 4.1 実装・実験内容

本稿では，第 3 章で提案した優先度制御通信手法における優先度通信フェーズの有効性を評価するため，実機上に実装を行った．評価実験に使用した機材を**表 6** に示す．IoT デバイスとして Raspberry Pi 2 を，AP サーバ，Broker サーバとして Ubuntu を使用した．

本実験ではモバイルネットワークを用いた IoT システムについて，アプリケーションの QoS を考慮した通信が可能かを検証するため，モバイルネットワークを介した送信実験を行った．評価実験の概要を**図 8** に示す．評価実験では 4 台の IoT デバイスからコンテンツデータを送信する．コンテンツデータの送信は優先度通信手法を適用した場合としない場合の 2 通りの方法で実施する．評価指標として①優先度別送信率（実験時間内に送信できたデータの割合），②優先度別平均遅延時間（IoT デバイスにデータが到着してから送信完了までの所要時間），③オーバヘッド送受信バイト数（制御メッセージの送受信に消費したバイト数）を用いる．①と②では優先度通信手法を用いない通信での送信と比較することで，提案手法がアプリケーションの QoS を考慮した通信が可能であるかを確認する．優先度通信手法を用いない場合と比べ，優先度の高いデータの送信率と平均遅延時間が改善している場合，提案手法はアプリケーションの QoS を考慮した通信が可能であると言える．③では，優先度制御通信手法に必要な制御メッセージのバイト数を確認する．この値がコンテンツデータに比べ十分小さければ，制御メッセージが通信に与える影響は微小であるため，優先度通信手法に関わるオーバヘッドは無視できると言える．

評価実験の送信シナリオを**表 7, 表 8** に示す．本シナリオは，研究室の他プロジェクトとして進んでいる農業支援システムの要件からシナリオを製作した．シナリオではカメラとセンサが 1 分に 1 回データを生成する．生成したデータのうち大容量の画像データは 1 分に 10 回高優先度（優先度 1～5）で，小容量の画像データとテキストデータは 5 分に 1 回高優先度で送信する．評価条件は**表 9** に示す．

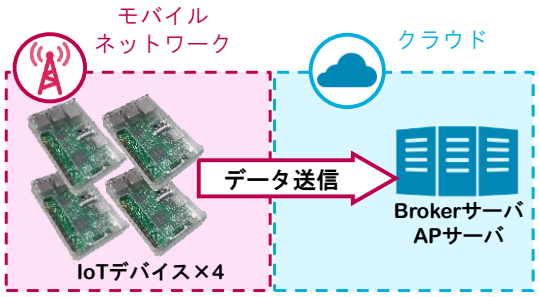


**図 8** 評価実験概要図

**表 6** 評価実験使用機材

| | IoT デバイス(Raspberry Pi 2) | AP サーバ,Broker サーバ |
|---|---|---|
| OS | Raspbian 8.0 | Ubuntu 14.04 LTS |
| CPU | ARM Cortex-A7 | Intel Corei7 5820K |
| メモリ | 1GB | 16 GB |
| 言語環境 | C++11(GCC4.8) + Boost1.55 | |

**表 7** 送信シナリオ(デバイス構成)

| DeviceID | デバイス名 | 送信データ量 | データ発生間隔 |
|---|---|---|---|
| 101 | カメラ (大: jpg，小: sjpg) | 2MB / 80KB | 1 分 |
| 102 | | | |
| 103 | センサ GW (温湿度/ 光量: txt) | 80B | |
| 104 | | | |

**表 8** 送信シナリオ(優先度決定テーブル)

| ColumnID | DeviceID | DataType | DataInterval | Priority |
|---|---|---|---|---|
| 1 | 101 | jpg | 10 分 | 2 |
| 2 | 101 | jpg | 1 分 | 7 |
| 3 | 101 | sjpg | 5 分 | 1 |
| 4 | 101 | sjpg | 1 分 | 6 |
| 5 | 102 | jpg | 10 分 | 3 |
| 6 | 102 | jpg | 1 分 | 8 |
| 7 | 102 | sjpg | 5 分 | 1 |
| 8 | 102 | sjpg | 1 分 | 6 |
| 9 | 103 | txt | 5 分 | 4 |
| 10 | 103 | txt | 1 分 | 9 |
| 11 | 104 | txt | 5 分 | 5 |
| 12 | 104 | txt | 1 分 | 10 |

**表 9** 評価条件

| | |
|---|---|
| 実験時間 | 1 回 60 分 |
| 使用ネットワーク | b-mobile(最高 128Kbps) |
| 実験場所 | 研究室内 |

### 4.2 実験結果・考察

本実験で得られた優先度別送信率を**図 9** に示す．優先度 1～5 の送信率は，平均で 36.7%，最大で 75.0%上昇した．さらに優先度が高くなるにつれ送信率も単調増加をしてい

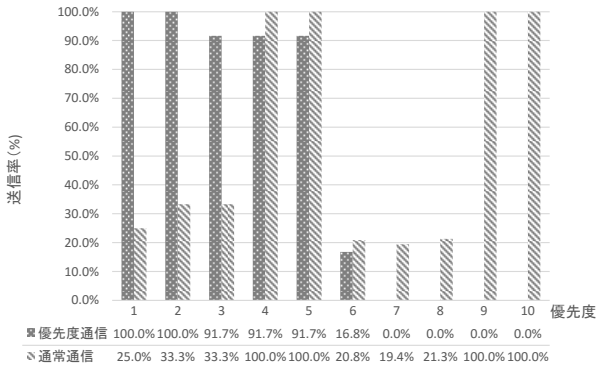


**図 9** 優先度別送信率

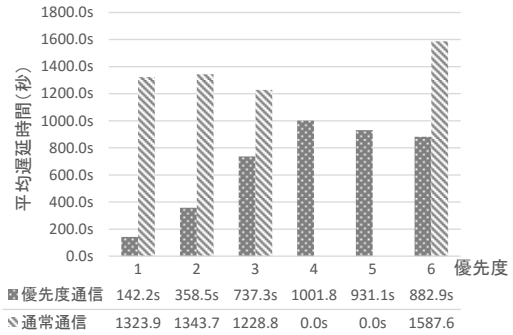


**図 10** 優先度別平均遅延時間

**表 10** 優先度通信手法のオーバヘッド

| DeviceID | 101 | 102 | 103 | 104 |
|---|---|---|---|---|
| 制御バイト数（A） | 2,513 | 2,509 | 1,218 | 1,218 |
| データバイト数（B） | 16,256,565 | 16,256,565 | 902 | 902 |
| A（コンテンツデータ 1 つあたり） | 21 | 21 | 20 | 20 |
| A（送信 1 回あたり） | 101 | 104 | 112 | 112 |

る．そのため，提案手法はアプリケーションの QoS を考慮して通信できていると言える．一方優先度が 4 と 5 のデータは，優先度通信を用いない方が高い送信率を示している．優先度通信を用いない場合，IoT デバイスは他の IoT デバイスの状況を加味した送信の遅延を行わないため，他の IoT デバイスが発生させた大容量データの遅延の影響を受けずに送信を開始できる．優先度が 4 と 5 のデータはセンサ GW からのテキストデータであるため送信開始・終了ともに全て成功したと考えられる．送信率が上昇した優先度 1～3 のデータに限定した場合，送信率は平均 66.7%上昇した．優先度別平均遅延時間を**図 10** に示す．図 10 は，優先度通信時にデータを送信することができなかった優先度 7～10 のデータに関しては集計対象外とし，優先度 1～6 の結果のみを示す．優先度が 4 と 5 のデータを除いた平均遅延時間は通常通信に比べ平均で 38.3%，最大で 10.7%まで削減できた．また平均遅延時間は優先度 1 のデータが最も短く，以降は増加する傾向にあるため高い優先度のデータをより短い遅延時間で送信できている．このためアプリケーションの QoS を考慮して通信しているといえる．優先度が 4 と 5 のデータは先述の送信率について議論した時と同様に，他のデバイスが発生させた大容量データの遅延の影響を受けず，加えて非常に軽量なテキストデータの送信であるため送信に要する時間が非常に短く，遅延時間が発生しなかったと考えられる．

優先度通信手法の制御メッセージによるオーバヘッド一覧を**表 10** に示す．制御メッセージに必要なバイト数はコンテンツデータ 1 つにつき 20 バイト程度であり，大容量データに対しては十分小さい．テキストデータの送信では十分小さいとは言えないが，制御メッセージの送信量より少ない容量のテキストデータは限られるため，優先度通信手法を使用することはできると考える．ただしテキストデータの送信率が低い場合，コンテンツデータより制御データのバイト数が多くなることがある．本実験では表 10 に示すように，センサ GW である DeviceID が 103,104 のデバイスは，送信制御に用いたバイト数が送信したコンテンツデータのバイト数を超えている．

以上から優先度通信手法は多種多様なデバイスがデータを送信するリモートモニタリング IoT システムでは有効に機能すると考えられる．ただし優先度が高くない場合，センサ GW などの軽量なテキストを送信するデバイスに対しては十分な効果を得ることができない場合がある．そのため軽量なテキストデータの送信を改善するデータ選択アルゴリズムをさらに検討する必要があり，本項目は今後の課題とする．

## 5. おわりに

本研究では，リモートモニタリング IoT システムにおける送信データの削減を行う通信制御の実現に向け，IoT デバイスが送信する多種多様なデータの持つ優先度に着目し，アプリケーションの要求に応じて適切に送信データの優先度制御を行う IoT 環境向け優先度制御通信手法を提案した．さらに，優先度制御通信手法の優先度通信フェーズについて実装と評価を行った．評価の結果高優先度の 1～5 の送信率について，平均で 36.7%，最大で 75.0%の送信率の向上がみられ，優先度 4，5 を除いた平均遅延時間は平均で 38.3%，最大で 10.7%まで削減できた．今後は，データ選択アルゴリズムの改良を行うほか，実際の農業支援システムへ導入しての評価を予定している．

## 6. 謝辞

本研究の一部は，科学研究費補助金基盤研究 B（26280028）により実施したものである．

## 参考文献


[1] Towards a Definition of the Internet of Things (IoT), IEEE, 2016(オンライン)，入手先 <http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf> (参照 2016-02-15）.

[2] Atzori, et al. “The internet of things: A survey.” Computer networks 54.15 pp.2787-2805, 2010.

[3] Gubbi, et al. “Internet of Things (IoT): A vision, architectural elements, and future directions.” Future Generation Computer Systems 29.7 pp.1645-1660, 2013.

[4] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019. Technical report, Cisco, 2015.

[5] SOLACOM, SORACOM INC(オンライン), 入手先 <https://soracom.jp/> (参照 2016-02-15）.

[6] Zhengguo, et al. “Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT.” IEEE Access 3 pp.622-637, 2015.

[7] Joseph. “The Story of the Internet of Things: Issues in utility, connectivity, and security.” IEEE Consumer Electronics Magazine 4.4 pp.54-61, 2015.

[8] Zorzi, et al. “From today's intranet of things to a future internet of things: a wireless-and mobility-related view.” IEEE Wireless Communications 17.6 pp.44-51, 2010.

[9] Palattella, et al. “Standardized protocol stack for the internet of (important) things.” IEEE Communications Surveys & Tutorials 15.3 pp.1389-1406, 2013.

[10] Sheng, Zhengguo, et al. “A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities.” IEEE Wireless Communications 20.6 pp.91-98, 2013.

[11] MQ Telemetry Transport, MQTT.org(オンライン),入手先<http://mqtt.org/> (参照 2015-4-1).

[12] 西岡哲朗, 他, “モバイルデータオフローディングプロトコル（MDOP）の提案, ” 情報処理学会 DICOMO シンポジウム, pp. 613-620, 2014.

[13] Shelby, et al. “The constrained application protocol (CoAP).” 2014.

[14] 3GPP TR 23.888, “System improvements for Machine-Type Communications （MTC）, ” 2013.

[15] Hirofumi Ibayashi, et al, “Highly Reliable Wireless Control System for Tomato Nutriculture,” IEEE 4th Conference on Consumer Electronics (GCCE2015), .2015.

[16] 柴田瞬, 他, “Optical Flow を用いた植物水分ストレス推定手法の検討, ” 第 78 回情報処理学会全国大会, pp.2-335－2-336 (2016.3).