Title	Internet of Thingsを対象とした大規模実証実験環境構築に関する研究
Author(s)	岩橋，紘司
Citation	
Issue Date	2015-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/12664
Rights	
Description	Supervisor:篠田陽一，情報科学研究科，修士

修 士 論 文

Internet of Thingsを対象とした大規模実証実験環境構築に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

岩橋 紘司

2015 年 3 月

修 士 論 文

Internet of Thingsを対象とした大規模実証実験環境構築に関する研究

指導教員　篠田 陽一 教授

審査委員主査　篠田 陽一 教授
審査委員　丹 康雄 教授
審査委員　知念 賢一 准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

1210011 岩橋 紘司

提出年月: 2015 年 2 月

概 要

x86、SPARC、Power などのサーバ用途またはデスクトップ用途のアーキテクチャで構成された、一般的な計算機以外の「モノ」をインターネットに接続し、情報収集や機器制御を行う、Internet of Things(モノのインターネット、IoT) が実現可能となりつつある。IoT を実現可能にしつつある要素技術は様々あるが、小型プロセッサや低消費電力の無線通信方式の発展が主な背景である。IoT で用いられるセンサやアクチュエータなど、限定的な機能を提供するための限定的な計算リソースを有する計算機を、一般的な計算機と区別して IoT デバイスと呼ぶ。IoT のネットワークシステムは、ワイヤレスセンサネットワークをはじめ、IoT デバイスを大規模に分散配置するネットワークシステムである。このようなネットワークシステムは、システムの展開後に、IoT デバイスの回収を伴うシステムの更改が難しい。そのため、他のネットワークシステムと同様かそれ以上に、システム展開前の段階における十分な検証が重要である。IoT の実証実験の要件は、自由なネットワーク構成、異なる計算機アーキテクチャの混在、通信メディアの混在、物理的な諸要素の制御を行う機構である。さらに、大規模実証実験においては、これらに併せてスケーラビリティも要件である。IoT のネットワークシステムの検証を行うための既存手法は 2 つに大別出来る。一つは、実際に用いる IoT デバイスを拠点に集約したテストベッドを用いる手法である。この手法の場合、拠点の空間的制約があり、集約出来る IoT デバイス数や無線通信を実験する空間の大きさに限界がある。もう一つは、IoT デバイスを模倣し、シミュレータ上のネットワークに配置することで、IoT デバイスによるネットワークをシミュレーションする手法である。シミュレータの実装にもよるが、シミュレーションを行う計算機の性能に制約を受ける。また、シミュレータ外部のネットワークとの通信も限定される。どちらの既存手法においても、前掲の要件を十分に満足することは難しい。特に、大規模実証実験を行う場合、実験の規模の拡大が困難である。

ネットワークシステムの実証実験を行うための施設としてテストベッドが存在する。テストベッドには、インターネットを利用したテストベッドと、インターネットから独立したテストベッドの 2 種類が存在する。IoT のネットワークシステムにおいては、無線通信の品質が重要なパラメタとなる。パラメタの制御を行うためには、外乱の排除が必要である。したがって、刻々と変化するインターネットの通信品質の影響を受けないインターネットから独立したテストベッドが望ましい。StarBED 型テストベッドはインターネットから独立した実験ネットワークを有するテストベッドである。インターネットや他の実験者のネットワークから独立した実験ネットワークを利用することで、通信品質に対する外乱を排除することができる。しかし、StarBED 型テストベッドは一般的な計算機による実験ノードと有線のネットワークで構築された計算機クラスタである。そのため、一般的な計算機と異なるアーキテクチャで実装された IoT デバイスのアプリケーションをそのまま実行することが出来ない。また、有線のネットワークにおいて無線ネットワークの技術をそのまま利用することはできない。

本研究は、StarBED 型テストベッドを用いて IoT を対象とした大規模実証実験環境を構築するためのフレームワーク Generic Utilization of Assorted Networking (GUAN) を提案した。GUAN の設計において、2 つの構成要素を取り入れる必要があった。一つは、StarBED 型テストベッドにおいて IoT デバイスのアプリケーションを対象にした実験を可能にするための構成要素である。そのため、IoT デバイスで動作するアプリケーションを一般的な計算機上で実行可能にする IoT デバイスの模倣手法を仮想 IoT デバイスを定義し、フレームワークの中心に据えた。仮想 IoT デバイスを用いることで、StarBED 型テストベッドにおいて IoT デバイスで動作するアプリケーションの実証実験が可能となった。もう一つは、有線のネットワーク環境において、無線通信プロトコルによる通信を扱う構成要素である。GUAN は、仮想 IoT デバイスのインタフェース調整と通信の中継を行うフレームワークであり、仮想 IoT デバイスが出力するフレームを中継する。このため、無線通信プロトコルによるフレームも有線のネットワーク環境で扱うことができる。さらに、通信を中継する際に無線通信の品質変化を模倣することによって、擬似的に作り出した無線ネットワークにおいて仮想 IoT デバイスを通信させることができる。StarBED 型テストベッドにおいては、無線通信の品質変化を模倣する手法が研究開発されている。無線ネットワークエミュレータの Meteor はデータリンク層で動作するため、通信メディアの混在を要件とする IoT の実証実験においても有用である。したがって、これらの構成要素を取り入れた設計をすることで、通信品質の模倣も含めたネットワーク実験が可能となるフレームワークを実現した。

GUAN に基いて実験を行うために、仮想 IoT デバイスとアプリケーション、一意な識別子を有する仮想 IoT デバイスを大量に生成する機構、仮想 IoT デバイスの統一的な制御を行う機構、仮想 IoT デバイスの通信を中継する機構を実装した。これらを用いて StarBED において実験を行った。実験の結果から、複数の計算機を用いて実験規模を自由に拡大するという、本研究における提案の基本概念を実証した。その結果、GUAN は IoT の大規模実証実験において有用なフレームワークであると結論づけた。

既存の IoT の実証実験手法では、大規模実証実験を行うための規模の拡大が困難であった。本研究は、複数の計算機と仮想 IoT デバイスを用いて、より大規模な IoT の実証実験を行うフレームワークを提案し、実験を行い実証した。また、IoT の大規模実証実験における自由なネットワーク構成、計算機アーキテクチャの混在、通信メディアの混在、スケーラビリティの各要件に関して、GUAN は満たしていると結論づけた。この成果は、既存の IoT デバイスを集約するテストベッドや 1 台の計算機の上で実行するシミュレータでは困難であった実証実験の規模の拡大を容易にする。これによって、複数のネットワークシステムが協調動作するような多数の IoT デバイスによるネットワークシステムの実証実験が可能となり、IoT の研究開発に寄与する。

目 次

図目次

表目次

第1章　序論

1.1　背景: Internet of Things

Internet of Things(モノのインターネット、以下 IoT と略記) という概念が注目を集めている。Kevin Ashton の記事 [7] によると 1999 年が初出であるが、PC やワークステーション、サーバのような計算機以外の「モノ」をインターネットに接続し、情報の収集や機器の制御などを行う概念である。[1] 本節では、IoT において用いられる「モノ」やそれを用いたネットワークについて概観する。

1.1.1　「モノ」の定義

はじめに、「モノ」とは何かを概観する。Vasseur と Dankels [13] によれば、IoT における「モノ」とはスマートオブジェクトである。スマートオブジェクトの構成要素は 3 点である。センサやアクチュエータなど外部環境との物理的な相互作用に利用される情報の入出力機構、計算能力、通信能力であり、これに電源や補助記憶装置などが付随する。

「モノ」の構成はアプリケーション指向で選択されるため、PC のように汎用的な計算機である必要はない。つまり、「モノ」の構成は一様でなく、最低限の構成要素が共通しているにすぎない。ここでは、最低限な共通性を有する部分と、その能力を担う構成要素を挙げる。

計算能力

基本的にマイクロプロセッサ、マイクロコントローラユニット (MCU) によって担われる。製品としては MSP430 や AVR などが存在する。

通信能力

L1 は有線または無線。 複数のネットワークインタフェースを有する「モノ」もある。L2 は IEEE 802.3、IEEE 802.11 (Wi-Fi)、IEEE 802.15.4 など。

また、USB 端子を利用した SLIP による通信を行える製品もある。

物理的な相互作用

どのような相互作用を行うかはデバイスに依存する。ここでは、一例を挙げる。

[1] より抽象的な概念としてテレマティクス、部分的に共通する概念としてサイバーフィジカル、M2M、ユビキタス、アンビエントインテリジェンスなどがある。

センサ

温度、湿度、照度、位置情報、音声、画像など外部の物理的な情報を取得する。

アクチュエータ

アクチュエータ、LED など、外部に物理的な作用を行う。

これらの構成要素から成る「モノ」とは、センサやアクチュエータなど限定的な機能を提供するための限定的な計算リソースと通信機能を有する計算機と定義できる。

1.1.2 「モノ」のネットワーク

ユビキタスコンピューティングやアンビエントインテリジェンスなど、身の回りにある「モノ」が生成する情報を収集、処理そして提供するようなネットワークシステムの概念は以前より存在している。ただし、このようなネットワークシステムにおいては「モノ」に込められる計算能力の制約が大きく、Internet Protocol (IP) を利用することが容易でなかった。また、一般的に「モノ」のネットワークシステムはアプリケーション指向で構築される。そのため、1 つのネットワークシステム上のエンドノードが他のネットワークシステムやそのネットワークのエンドノードと通信する必要性も希薄であった。

1.1.3 「モノ」のインターネット：Internet of Things

ワイヤレスセンサネットワークなどの「モノ」のネットワークは従来、L2 で用をなしていた。しかし、IoT においては L3 に IP を利用する。L3 が使えることで L2 によるネットワークと比較してどのような特性を得られるのか概観する。

一般に、L2 の通信はブロードキャストであり、「モノ」のネットワークは単一のブロードキャストドメインで構築される。他方で、L3 を使うことで、L2 的なブロードキャストドメインを分割できる。つまり、ネットワークという単位で区別して通信が可能となる。

歴史的に L3 のプロトコルとして IPX や AppleTalk などが利用されていたが、現在、L3 のプロトコルは IP を除いたプロトコルは現実的に利用されていない。ネットワークプロトコルスタックは、L3 の IP を中心にいわゆるナローウェストモデルである。IP は上位層・下位層のネットワークプロトコルを問わない。すなわち、IP を処理することができれば、インターネットを通じて別のネットワークのエンドノードと通信することが可能となる。ワイヤレスセンサネットワークなどのネットワークシステムが採用する L1、L2 の違いを超えて、それらのネットワークシステムを相互接続することが可能になる。

しかしながら、「モノ」のネットワーク間における通信に IP を利用する上での問題点が存在する。IPv4 の場合はアドレスの枯渇問題に直面しており、新たに「モノ」にグローバル IPv4 アドレスを割り当てることは現実的でない。また、IPv6 の場合はプロトコルが要求する MTU と比較して MTU がより小さなデータリンク層のプロトコルを利用することができなかった。特に、「モノ」のネットワークで用いられる低データレート、低消費電力

の無線通信プロトコルである IEEE802.15.4 の MTU は 127byte しかないため、IPv6 が要求する最低 1280byte の MTU に満たない。しかし、MTU の小さな L2 でも 1280byte の MTU を要求する IPv6 を利用可能にするための、6LoWPAN ワーキンググループによるヘッダ圧縮機構 [9] をはじめとする、手法が発展したため、L3 に IP を用いることが出来るようになった。これにより、IP によって相互に接続されたネットワークからなる現状のインターネットに「モノ」のネットワークが参加することが可能になったのである。

IoT の定義　Vasseur と Dunkels[13] によるとスマートオブジェクト・テクノロジ (Smart object technology) は、IoT や the web of objects、the web of thihngs、cooperative objects など他の類似の概念と含意や定義に微細な違いはあるにせよ、基本的な技術は同様とだとした上で、スマートオブジェクト・テクノロジと呼称している。IoT はスマートオブジェクトによる「モノのネットワーク」が IP を用いて通信することを可能にしたネットワークシステムと解することが出来る。これに加えて、小文字で表記した internet の意味する相互接続網という性質を強調すべきである。「モノ」による internet は「モノ」によるネットワークシステムの間で相互接続がなされ、個々の「モノ」が一意な識別子を割り当てられる。つまり、IoT の概念の説明においては、スマートオブジェクトが IP を用いて通信することを可能にしたネットワークシステムであるのみではなく、IP を用いることによって得られる特性である、スマートオブジェクトによって構築されたネットワークシステムのエンドノードに対してもインターネットを通じた通信が可能になったことを強調すべきである。したがって、IoT とは、IP を用いて通信する「モノ」のネットワークが相互接続することによってネットワークシステムを構成する概念である。

IoT においては、ユビキタスコンピューティングやアンビエントインテリジェンス、センサネットワークなどのネットワークシステムにおけるエンドノード及びネットワーク間の通信に IP を用いる。つまり、ワイヤレスセンサネットワークやアドホックネットワーク、遠隔計測、ファクトリーオートメーションなど、従来、IP を用いずに構築されてきたネットワークにおいても IP を用いて通信するネットワークとして構築可能になる。各ネットワークにゲートウェイを設置することで、インターネットを通じてこれらのネットワークの相互接続が実現する。

IoT に対応したデバイスやネットワークシステムが普及すると、インターネットを構成するエンドノード、ネットワークの種類が増えることが予見される。ネットワークには、Low-rate/power Lossy Network (LLN) と呼ばれる低レート、低出力、ロスの多い、といった特徴を有するネットワークも含まれる。TCP/IP を用いたネットワークでほぼ前提とされてきた安定した接続性が確保されないネットワークもインターネットの一部になることが予見される。[2]IoT では、PC と異なるアーキテクチャを採用した「モノ」や、LLN での利用を想定したネットワーク技術が用いられる。ハードウェアのアーキテクチャとネットワーク技術の両方の面おいて多様性が増加しつつある。

[2]LLN に類似の概念で特に遅延や途絶、切断に対する耐性を有するネットワーク DTN(Delay/Disruption/Disconnection Tolerant Network) がある。DTN を主眼に据えた研究領域がある。

1.2　本研究の目的

IoT のネットワークシステムの大きな特徴として、センサなどの小規模な「モノ」もインターネットに接続されることが挙げられる。大規模な普及、展開を前提に開発される「モノ」のアプリケーションは、設計の段階で水平スケールすることに加え、十分な実証実験が求められる。しかしながら、対象のデバイスを用いて実験するには、調達費用や空間的な制約など課題が多い。無線通信を利用するデバイスの場合、外乱の影響も考慮に入れる必要がある。また、実証実験における、再現性、可制御性、規模追従性の要請は他のネットワークシステムと同様に存在する。

本研究は、IoT を対象とした実証実験を行う上での要件を整理し、それを満たす方法の確立を目的とする。

1.3　本論文の構成

本論文の構成は以下の通りである。第 2 章では、IoT の要素技術について概観する。第 3.3.2 章では、ネットワークシステムの大規模実証実験について概観する。第 4 章では、IoT の実証実験について概観し要件を定義した上で、IoT を対象に大規模実証実験を行うために既存技術の利用を検討する。第 5 章では、IoT を対象とした大規模実証実験フレームワーク GUAN を提案する。第 6 章では、GUAN の実装例を提示し第 7 章において、大規模実証実験例を報告する。第 8 章では、実験から得られた結果を評価し、また、IoT の大規模実証実験に関して本研究の採用した方法の妥当性を議論する。

第2章　IoTと要素技術

本章では Internet of Things(モノのインターネット、IoT) の要素技術について概観する。

2.1　ハードウェア

IoT において用いられるハードウェアは多岐にわたる。PC やタブレット型端末をはじめとする、ユーザとの直接的な相互作用を前提とした端末の他にも、家電、センサなど多様なハードウェアの利用が想定される。以下では、センサやアクチュエータなどの限定的機能を提供するための限定的な計算リソースを有する計算機を、PC を始めとする一般的な計算機と区別して IoT デバイスと呼ぶ。

IoT デバイスのハードウェア的構成は、一般にマイクロコントローラ、ネットワークインタフェース、センサやアクチュエータ等の物理的相互作用のための要素、記憶装置、電源が含まれる。

マイクロコントローラ

MSP430、AVR、MIPS、PIC など。一般的な PC で用いられる x86 のプロセッサと異なる命令セットを持つアーキテクチャが採用されることもある。

Atmel、Freescale、Texas Instruments などのマイクロコントローラメーカーをはじめ Intel や Qualcomm、NVIDIA などチップメーカーも小型プロセッサの開発に注力している。

通信インタフェース

IEEE 802.3 (Ethernet)、IEEE 802.11 (Wi-Fi)、IEEE 802.15.4、IEEE 802.15.1 (Bluetooth) などのインタフェースが不可欠である。USB を利用した SLIP 通信が利用可能な IoT デバイスも存在する。

通信の形態、プロトコルについては 2.3 節にて説明する。

物理的な相互作用のための要素

温度、湿度、照度、音声、画像、位置情報などのセンサ類やアクチュエータ、LED など。

記憶装置

オンボードのフラッシュメモリ、SD カード等。

電源

IoT においては、電池による長期間の動作を期待されるデバイスも存在する。

2.2 オペレーティングシステム

IoT デバイス向けの OS には、一般的な PC やモバイル端末向けの OS と異なる OS も用いられる。Contiki-OS や TinyOS、FreeRTOS など、センサノードを始めとする低資源なプラットフォームのために特化した OS がある。

また、汎用 OS を用いない組み込みシステムも想定される。

2.3 通信メディアとネットワークプロトコル

本節では IoT において使用されうる通信メディアとネットワークプロトコルについて概観する。

2.3.1 L1、L2 の要素技術

IoT では L3 において IP が用いられるが、その下のレイヤにおいて考慮すべき要素として L1、L2 の要素技術を概観する。IoT では、TCP/IP がほぼ前提としてきた安定した接続性を必ずしも保証できない L1/L2 で Low-rate/Low-power Lossy Network (LLN) を考慮する必要がある。LLN や LR-WPAN、LoWPAN という言葉で表現される低レート、低消費電力なネットワークをターゲットにした通信技術を用いてその上位層に IP を使用するという IoT デバイス特有の利用法についてもここで概観する。

L1 の要素技術

物理層で用いられる技術は一般に有線か無線である。有線の場合、RJ45 端子を扱える Netduino や Raspberry Pi などの小型計算機が存在する。他には、TelosB のように USB 端子を通じて SLIP を利用した通信が出来る製品も存在する。有線のインタフェースを持つ IoT デバイスの場合、Power over Ethernet (PoE) による給電など、一石二鳥の利用方法も想定できる。無線の場合、主な周波数帯は 900MHz 帯、2.4GHz 帯および 5GHz 帯が用いられる。

L2 の要素技術

IEEE 802.3 いわゆるイーサネットである。

IEEE 802.11 Wi-Fi と必ずしも同値じゃないけど。喋れるデバイスは安定した電力供給があるか頻繁に充電されるようなブツか。

IEEE 802.15.4 IEEE 802.15.4 は Low-Rate Wireless Personal Area Network (LR-WPAN) の規格である。低消費電力の無線通信プロトコルであり、e や g など特定の用途に特化した拡張が存在する。また、ZigBee におけるデータリンク層にも用いられる。

IEEE 802.15.1 一般には Bluetooth として知られる。Bluetooth 4.2 より IPv6 への対応が可能となった。

2.3.2 L3 の要素技術

uIP、lwIP

計算リソースの乏しい IoT デバイスでも用いることができるプロトコルスタックが存在する uIP(micro IP) は 8bit や 16bit のマイクロコントローラでも扱える TCP/IP スタックである。現在は Contiki OS に統合されて用いられている。lwIP(lightweight IP) は組み込みシステム向けに設計された TCP/IP スタックである。

6LoWPAN によるヘッダ圧縮機構

6LoWPAN とは、IPv6 over Low power Wireless Personal Area Networks のアクロニムであり、原義は IETF の WG の名称である。原義はワーキンググループの名称であるが、そこで策定された IPv6 データグラムを IEEE802.15.4 ベースのネットワーク上で配送するためのヘッダ圧縮フォーマットや IEEE 802.15.4 と IPv6 の間に Adaptation Layer のことを指して 6LoWPAN と呼ぶ例も多い。問題報告は RFC4919[10]、基本的な特徴は RFC6282[9](RFC4944[12] の更新) で記述されている。

IP はどんなに小さなデバイスであっても適用できて、適用すべきであるという概念のもとで、6LoWPAN において定義されるのはカプセル化とヘッダ圧縮機構である。これらによって、IPv6 データグラムを IEEE 802.15.4 ベースのネットワークで送受信可能にする。

6LoWPAN は L3 の要素技術であるが、L2 の識別子に対する依存をはじめとする幾つかの前提としている機能がある。MTU が 127byte しかない IEEE 802.15.4 上で最小 MTU として 1280byte を要求する IPv6 を利用可能にするために Adapting Layer の使用が RFC4944 において提案されている。IEEE 802.15.4 ベースのネットワークに 6LoWPAN のボーダールータを用いることで、一般的なインターネットとの相互接続性を確保できる。

RPL

Routing Protocol for Lower Lossy Network のアクロニムであり、RFC6550 において定義されている。

2.4　開発におけるエミュレータの役割

本節では、PC と異なるアーキテクチャで実装されたハードウェアを用いるネットワークシステムの開発に際して用いられる方法論の 1 つとして、エミュレータを用いた開発の流れを説明する。PC と異なるアーキテクチャで実装されたハードウェアを用いるネットワークシステムの特徴として、開発環境として用いる端末と実際にシステムを展開する環境の間に、ハードウェアアーキテクチャの差異が存在する。そのため、設計から実際にシステム開発の段階へ移行する際に取りうる方法は 2 つに大別される。一つは、設計の段階でハードウェアと OS の組み合わせの選択が明確に決定した上で開発用のハードウェアを調達する方法であり、もう一つは開発の段階においてエミュレータを使用する方法である。

IoT のネットワークシステムの開発においてエミュレータを使用する利点がある。IoT のネットワークシステムにおいては要素技術やハードウェアの組み合わせが多岐にわたるが、エミュレータを利用することで実際のハードウェアを調達する前にアプリケーションの大まかな動作を予め確認することが可能である。エミュレータ上で動くコードはほぼそのまま実機にデプロイ可能である。

図 2.1 はエミュレータを利用した開発サイクルを表している。はじめに、システムを考案すると、仕様を策定し、設計が行われる。この段階ではハードウェア及び OS に対する要件を仕様に合うハードウェア及び OS を選択し、アプリケーションソフトウェアを作成する。エミュレータを利用してこのアプリケーションソフトウェアを検証し、その評価を元に変更が必要であれば、システムの設計やハードウェア及び OS の選択、アプリケーションソフトウェアの作成の各段階に戻って変更を加えることが可能である。この手順を繰り返し十分な評価が得られた段階で、実際のハードウェアを利用してシステムを展開することが出来る。

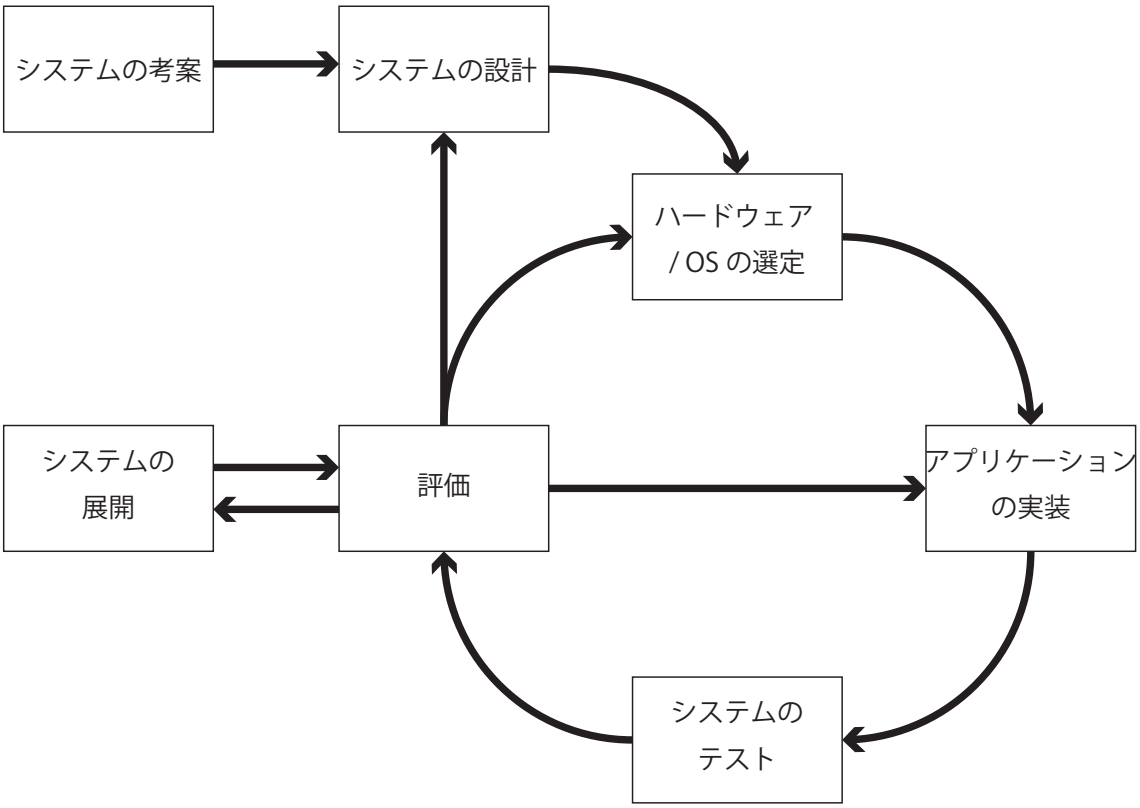

図 2.1: エミュレータを利用した開発サイクル

第3章　ネットワークシステムの大規模実証実験

本章では、ネットワークシステムの実証実験について概観する。そもそも何を対象に検証を行うのか、どのように検証を行うのか、どのような施設で検証を行うのか説明する。

3.1　検証の対象

本節ではネットワークシステムの検証の対象について説明する。検証の対象は、アプリケーションソフトウェアとネットワーク通信の 2 つに分けて説明する。

3.1.1　アプリケーションソフトウェア

ネットワークシステムの検証の基本的な対象の一つはアプリケーションソフトウェアである。検証では、作成されたアプリケーションが設計どおりに動作するか様々なパラメタを変更しつつ確認する。

物理的な情報に関連した動作

アプリケーションソフトウェアの検証に関して、IoT、サイバーフィジカルなどで用いられる、計算機外部から人の手を介さずに情報を取得したり、人の手を介さずに物理的な制御が行われるアプリケーションを考慮に入れるべきである。しかしながら、物理的な情報に関連した動作を検証するためには、外乱や周辺環境の物理的特性など、計算機の中だけで制御しきれない諸要素が存在する。それらの諸要素を実証実験においてどのように取り扱うか課題が存在する。

3.1.2　ネットワーク通信

アプリケーションソフトウェアの動作の検証と並んでアプリケーションソフトウェアによるネットワーク通信も検証の対象である。主に検証すべき要素はジッタ、レイテンシ、帯域である。ネットワークシステムが対応できる規模とそのためにネットワークに対して求められる性能をはじめとする要件を検証することができる。

3.2 検証の手法

アプリケーションソフトウェアを検証する上で、どのように検証するかも課題の一つである。本節では検証の手法について概観する。

3.2.1 物理マシンを用いた検証

アプリケーションソフトウェアの検証において最も確実な結果が期待出来る方法は、物理マシンを用いた検証である。この手法では、設計に基いて、実際のシステムにおいて利用する物理マシンと同様の構成を採った物理マシンでアプリケーションを動作させる。これによって、設計どおりの動作が確認できれば、実際に展開したアプリケーションソフトウェアで同様の動作が期待出来る。ただし、複数の物理マシンを必要とする場合は、必要台数確保する必要がある。

3.2.2 仮想マシンを用いた検証

検証において、仮想マシンを用いることで 1 台の物理マシンで複数のマシンを動作させることも可能である。ハードウェアの数が限定されるが複数のハードウェアを用いて検証したい、という場合に仮想マシンを用いて多重化を行うことで実験に用いるマシンの数を増やすことが可能である。ただし、仮想マシンの性質上、CPU やネットワークインタフェースなどの資源を使用する際に他の仮想マシンの処理を待たなくてはならない場合も発生しうる。

3.2.3 物理的な情報の検証

前節において、物理的な情報に関連した動作も検証の対象となりうることを論じた。物理的な情報をシミュレーションし、実験対象のアプリケーションの入出力にシミュレーション結果を反映することで、物理的な情報を模倣することが出来る。物理的な情報を模倣することで、外乱を排除し実験の再現性を高めることが可能になる。ただし、モデル化の方法や精度に応じて検証結果の迫真性に差異が生じることに留意が必要である。

3.2.4 統合型シミュレータ

NS2 / NS3

NS (The Network Simulator)[1] はネットワーク研究のための離散イベントシミュレータである。有線および無線ネットワーク上での TCP、ルーティング、マルチキャストのシミュレーションをサポートしている。

ユビキタスネットワークシミュレーション環境 RUNE

RUNE (Real-time Ubiquitous Network Emulation Environment) [15] は計算機クラスタ環境におけるユビキタスネットワークのシミュレーションを支援するプラットフォームである。StarBED など PC で利用するネットワークシステムのアプリケーションを対象とした検証に有利なテストベッドにおいて、RUNE は大規模なユビキタスネットワークシステムのエミュレーションを補助する。ユビキタスネットワークシステムのエミュレーションに関連する RUNE の主要な特徴は、実時間での動作、マルチレベルのエミュレーションレイヤ、温度、湿度、照度など周辺環境のエミュレーション能力の 3 点である。

3.3 実証実験施設

本節では、ネットワークシステムの実証実験を行う施設や設備として機能する枠組みについて概観する。主な形態として 2 種類あり、インターネットを利用した実証実験施設とインターネットから独立した実証実験施設がある。本論文では本節以降、実証実験施設や設備として機能する枠組みをテストベッドと呼ぶ。

3.3.1 インターネットを利用したテストベッド

インターネットを利用したテストベッドとは、複数の拠点が資源を提供し、インターネット上にオーバーレイネットワークを構築し実験を行うテストベッドである。このようなテストベッドにおいては、実際のインターネットを用いるため、帯域や遅延などインターネット上の通信に関連する諸要素の影響を受けた迫真性の高いネットワーク実証実験が可能である。他方で、実際のインターネット上で刻々と変化する帯域、遅延など制御が困難な諸要素の影響を受けるため、実験の再現性に関して限界がある。この形態のテストベッドは複数の拠点が資源を提供するため、一拠点が用意する資源が限られていても実験の規模の制約を受けにくい特徴がある。

PlanetLab

PlanetLab[2] は新しいネットワークサービスの開発を支援するグローバルに分散した研究ネットワークである。テストベッドとしての PlanetLab はグローバルに分散した計算機の集合であり、すべての計算機はインターネットに接続されている。また、テストベッドのすべての計算機上で動作している Linux をベースにした OS が動作し、MyPLC というソフトウェア・パッケージが存在する。また、MyPLC はダウンロード可能であり、任意の環境に MyPLC を用いて PlanetLab と同様のテストベッドを構築することが可能である。PlanetLab OS の変更が出来ないため検証できるアプリケーションが Linux 用の実装に限定される、root 権限が必要な操作に制限がある、などの制約がある。

GENI

GENI (Global Environment for Network Innovations) [3] は米国国立科学財団の「コンピュータとネットワークシステム」(Computer and Network System, CNS)[4] プロジェクトの一部である。GENI はネットワーク及び分散システムの研究及び教育のための仮想的な実験施設を提供する。

3.3.2 インターネットから独立したテストベッド

インターネットを利用したテストベッド以外にも、単一の拠点に計算機やネットワークの資源を集中させ、インターネットから独立したネットワーク環境を構築したテストベッドが存在する。このようなテストベッドでは、インターネットから独立したネットワーク環境であるため、刻々と変化するインターネット上の諸要素の影響を受けることなく実証実験が可能である。そのため、実験において制御が困難な諸要素が少なくなり、実験者は再現性の高い実験を行うことが可能である。

StarBED

StarBED Project[5] はインターネットの研究開発を支援する目的で大規模汎用インターネットシミュレーションが可能な常設のテストベッドを構築・運用している。施設としての StarBED は、情報通信研究機構テストベッド研究開発推進センターテストベッド研究開発室北陸 StarBED 技術センターのことを指し、1000 台を超える汎用 PC ノード、管理ネットワーク、他のネットワークから独立した実験ネットワーク、実験支援システムからなるネットワークテストベッドを提供している。

StarBED[11] はハードウェアとソフトウェアの両面から構成されている。ハードウェアとしては、複数のネットワークインタフェースを有する汎用 PC ノードが、実験ネットワークと管理ネットワークのスイッチにそれぞれ接続されている。実験ネットワークと管理ネットワークが分離されていることで、管理、制御に用いられる通信と実験で発生する通信を完全に分離することが可能である。ソフトウェアとしては、実験支援ソフトウェアの SpringOS がある。これらの構成要素を併せて構成される StarBED は、実験者に対して汎用 PC ノードや VLAN ID などの資源を提供し、任意の OS を用いたアプリケーションソフトウェアの実証実験を可能にする。

北陸 StarBED 技術センター以外の研究施設においても、汎用 PC ノードと必要に応じたネットワーク機器を用意することで StarBED と同様の構成を採用することが可能である。本論文では、StarBED と同様の構成を採用し、実験者に対して HaaS 的に資源を提供する実証実験環境を StarBED 型テストベッドと呼ぶ。

第4章　IoTの実証実験と既存技術利用の検討

IoTのネットワークシステムのなかでも特に、ワイヤレスセンサネットワークのような一旦デバイスを大規模分散配置した後にデバイスの回収を伴うシステムの更改が困難である。したがって、このようなネットワークシステムの開発においては、事前に十分な検証を行うことが必要である。

本章では、既存技術を概観し、既存技術の利用について検討する。さらに、既存技術で対応困難な課題を整理し、IoTの実証実験の要件について議論する。また、実証実験環境構築に利用する施設や要素技術の選択について議論する。

4.1　IoTの実証実験の要件

本節ではIoTの大規模実証実験を議論するにあたって、基礎となるIoTの実証実験についての要件を議論する。はじめに、IoTの実証実験はインターネットの実証実験であることが挙げられる。インターネットは複数のネットワークシステムの相互接続によって構成される。つまり、複数のネットワークシステムが相互に接続されたネットワーク構成を対象とした実証実験が可能であるべきである。インターネットの検証を離れて独自のネットワークのみ検証するのはセンサネットワークを始めとする従来的な「モノ」のネットワークの実証実験と同質なものとなってしまう。故に、IoTの実証実験環境は、実験者によって自由にネットワークが分割可能である必要がある。

次に、実証実験の対象となるネットワークシステムの多様性が挙げられる。つまり、センサネットワークやホームオートメーションなどの特定の目的のネットワークシステムのみを対象とした検証ではなく、ネットワークシステム一般を対象にした実証実験が可能であるべきである。このためには、ハードウェアアーキテクチャの混在と通信メディアの混在の2つの側面を考える必要がある。一つは、一般的な計算機と異なるアーキテクチャで実装されたIoTデバイスを使用することに起因する。もう一つは、L1、L2において用いられる要素技術は多様であることに起因する。IoTの多様な要素技術を利用したネットワークシステムを実証実験の対象として扱うためには、これらの混在を許容する必要がある。

さらに、物理的な操作及び周辺環境の諸要素の制御が挙げられる。これは、先に述べた通信メディアの多様性にも関連するが、物理的な周辺環境の諸要素は、無線通信において遅延や帯域など、通信品質の変化をもたらす。諸要素の制御ができれば、通信品質の変化

が制御可能となり、実験の可制御性、再現性につながる。また、センサのように物理的な情報を取得するIoTデバイスのためのアプリケーションを検証する場合には、入力される物理的な情報が制御可能であることで、実験の可制御性、再現性につながる。そのため、物理的な諸要素の制御が可能であることが望ましい。

IoTを対象とした実証実験環境に求められる要件について整理すると、通信メディアの混在およびハードウェアアーキテクチャの混在を許容し、物理的な諸要素の制御が可能で、自由なネットワーク構成が採用出来る必要がある。

4.2 既存のIoT実証実験手法

4.2.1 実際のIoTデバイスを用いた検証

一般的なネットワークシステムの研究開発と同様、IoTの実証実験においても最も実証的な方法は、システムを展開するため実際に利用するIoTデバイスを用いて検証することである。

実際のハードウェアを用いたテストベッド

ネットワークシステムの実証実験に用いる施設として、一拠点に資源を集中させたテストベッドが存在することは3.3.2において述べた。一般的なネットワークシステムの研究開発と同様に、IoTの研究開発においても、IoT-LAB[6]のような実際のIoTデバイスを多数配置し、実証実験を行うことが出来るテストベッドがある。IoT-LABのテストベッドは、各拠点にそれぞれ928、640、400、344、256、160のワイヤレスセンサノードがあり、各拠点でトポロジや環境が異なる。IoT-LABのテストベッドはそれらの拠点の合計2728ノードで構成される。

実際のIoTデバイスを用いることで、ソフトウェアの動作に関してもっとも現実的なデータを取得出来ることが期待出来る。

4.2.2 統合型ネットワークシミュレータ

IoTの研究開発を行っている組織において、ハードウェアエミュレータ、ネットワークシミュレータを組み合わせ、ネットワークシステムのシミュレーションを行う統合型のネットワークシミュレータを開発している例がある。

MSPSim/COOJA

MSPSimはMSP430シリーズのマイクロプロセッサとそれを利用した一部プラットフォームのエミュレータである。MSPSimのスタンドアロン版は1台のPC上で実行される複数のMSPSimの間の通信を実現することが可能である。ただし、その通

信に無線のエミュレーションを加えることは出来ず、すべての MSPSim 間で通信が可能なネットワークとなる。

COOJA はエミュレータを統合したネットワークシミュレータである。利用するエミュレータの動作も含んだシミュレーション上のイベントを管理するため、複数のエミュレータが協調した無線ネットワークのシミュレーションが可能である。しかし、COOJA は、シミュレーションの外に存在するネットワークと通信する手段が限定的である。COOJA 上のノードと通信する場合、COOJA を実行するホストが COOJA 上の特定のノードと SLIP で通信することで間接的に COOJA 上の他のノードと通信する。

また、COOJA 上において実時間相当で動作させられるノード数に限界がある。アプリケーションやノードの配置によるが、実時間相当で動作するノード数は 20 から 30 程度である。

WSim/WSNet

WSim/WSNet は SenseLab の開発するエミュレータとネットワークシミュレータである。WSim 単体で動作させる場合は実時間での動作も可能であるが、WSNet によるネットワークシミュレーションを利用する場合実時間での検証が出来ない。

WSNet は WSNet1 と WSNet2 の二種類があり、WSNet1 は MSPSim の Network Connection と同様に完全な通信手段によるフルメッシュのネットワークを構築する。WSNet2 はイベントドリブンなネットワークシミュレーションを行う。

4.2.3 既存技術で対応が困難な課題

IoT の実験において、実際の IoT デバイスを用い、実際に無線通信を行うことが最も実証的な実験であることは言うまでもない。しかし、実際の IoT デバイスで無線通信を行う実証実験の環境を構築するには、IoT デバイスの調達コストに加え、実験を行う空間を確保することが必要である。つまり、実際の IoT デバイスによる無線通信の実験はスケールアウトが容易でない。また、一旦 IoT デバイスを調達し配置すると、対象となる IoT デバイスの配置や構成を柔軟に変更することも難しい。さらに、無線通信に対する外乱などの物理的な諸要素を制御することも難しい。

シミュレーション上のイベントをすべて管理できる統合型ネットワークシミュレータは、実装によって実時間より早い動作で実験することも可能である。しかし、これらのシミュレータはシミュレーション上のイベントの管理を協調・分散して行う実装になっていない場合、複数の PC を利用して大規模な実証実験を行うことが不可能である。また、シミュレータの外部と通信する手段が備わっていない、あるいは限定的であり、外部のネットワークと通信することが困難であるため、実験に統合型シミュレータを用いる場合、他のネットワークシステムと協調し、多数の異なるネットワークと相互接続する実験を行うことも難しい。

4.3 IoTの大規模実証実験の要件

IoTの実証実験に関する要件は、第4.1節において予め論じた。IoTの実証実験に大規模という条件を加えた場合、先に論じた要件に加えて、実験規模の柔軟な拡張が可能となるスケーラビリティの確保が挙げられる。以下は、先に論じた要件とともに整理した、IoTの大規模実証実験の要件である。

自由なネットワーク構成

IoTは単一のネットワークシステムで収まらず、多様なネットワークシステムの相互接続によって構成される。このため、実証実験においても自由なネットワーク構成が可能である必要がある。

計算機アーキテクチャの混在

IoTデバイスの中には、電池による長期的な動作を期待されるセンサ類をはじめとする電源に制約のあるハードウェアが含まれる。これらのハードウェアは、x86と事なるアーキテクチャで実装される場合がある。実証実験においては、PCやIoTデバイスが混在するネットワークシステムも対象となる。このため、異種アーキテクチャの混在を許容出来る必要がある。

通信メディアの混在

IoTにおいて相互接続されるネットワークシステムは同じ通信メディアを用いていると限らない。そのため、一方は有線、他方は無線が用いられるネットワークシステムの相互接続について検証する場合、通信メディアの混在した環境を構成する必要がある。

物理的な諸要素の制御

IoTにおいては周辺環境から受ける物理的な諸要素によって無線通信における通信品質の変化が生じる。また、センサなど物理的な情報を取得するアプリケーションの実証実験においても重要となる。このため、IoTの実証実験においては物理的な諸要素の制御が必要となる。

スケーラビリティ

既存手法における問題点の一つは、実験可能な規模が制約される点である。多様なネットワークシステムからなるIoTを対象に実証実験を行ううえで、実験規模に制約を受けるのは望ましくない。このため、実験規模が柔軟に変更可能である必要がある。

4.4 実証実験施設の選択

本研究では、第4.3節において、IoTの大規模実証実験環境に対する要件を整理した。本節では、IoTの大規模実証実験環境構築のためにどのような実証実験施設が望ましいか

を検討する。

4.4.1　インターネットを利用したテストベッド

インターネットを利用し、その上にオーバレイネットワークを構築して実験を行うテストベッドにおいては、遅延や帯域など通信に関連した諸要素を制御できない。遅延や帯域は、無線通信を行う IoT デバイスで利用するアプリケーションソフトウェアにおいては重要なパラメタである。そのため、これらのパラメタを制御することが困難なインターネットを利用したテストベッドは、無線通信を行う IoT デバイスを含めたネットワークの実証実験を行う上で望ましくない。

4.4.2　インターネットから独立したテストベッド

インターネットから独立したテストベッドは、一箇所に資源を置いて実験を行うことで、刻々と変化するインターネットの遅延や帯域による通信品質への影響を受けずに実験を行うことが可能である。

StarBED 型テストベッド

StarBED 型テストベッドの基本的構成は先に第 3.3.2 章で説明した。StarBED 型テストベッドにおける一般的な実験は図 4.1 に示したように、複数の PC を実験ノードとして利用する実験ノードは管理ネットワークと実験ネットワークに接続される。実験者は管理ネットワークを通じて実験ノードの構築、制御を行う。実験ネットワークは、実際のインターネットや他の実験者から独立しており、実験ネットワークにおいては実験に関する通信のみが行われる。加えて、実験者は、割り当てられた VLAN ID の範囲内で、実験ネットワークを自由に分けることが可能である。また、StarBED 型テストベッドは、SpringOS によって実験環境の構築や実験遂行の支援が受けられる。特に、ディスクイメージの作成、配布によって実験ノードの複製が容易であり、実験ノードの追加によるスケーラビリティの確保が容易に行える。

また、QOMET、METEOR によって有線ネットワークで構築された計算機クラスタ上における無線ネットワークの通信品質を模倣することも可能である。加えて、RUNE によるユビキタスネットワークを対象とした実証実験の実績も存在する。

さらに、StarBED 型テストベッドは実験の駆動においても優れた点を有している。StarBED 型テストベッドでは実験の駆動において、Kuroyuri[16] によるスクリプト実行での実験駆動が可能である。スクリプト実行による実験駆動によって、実験者は実験の可制御性、再現性を確保可能である。

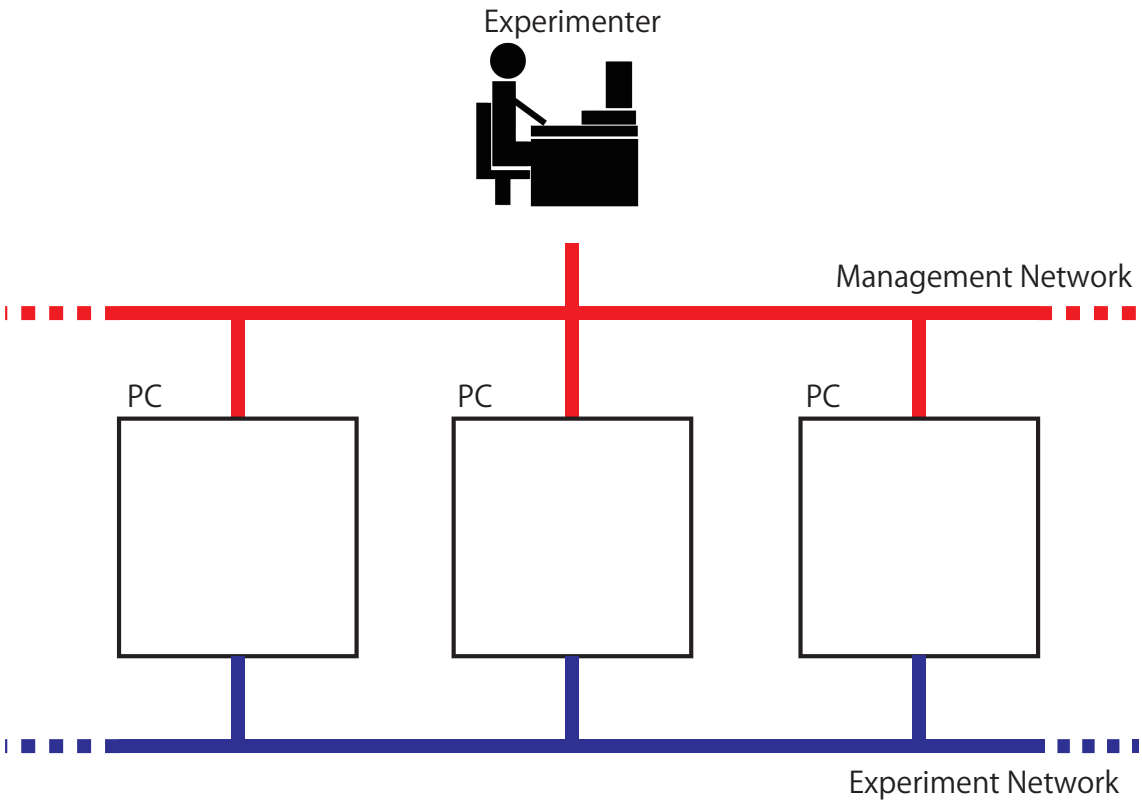

図 4.1: StarBED 型テストベッドにおける実験環境の基本的な構成

IoTを対象にした大規模実証環境の構築にはインターネットから独立したテストベッドが望ましい。自由な構成のネットワークを構築可能であるという点で、計算機のOSをはじめとする要素に制約を受けず利用可能なテストベッドが望ましい。異種アーキテクチャの混在については、実際のIoTデバイスをその都度調達する事は困難である。したがって、汎用の計算機上でアーキテクチャの違いを克服する方法を用意する必要がある。また、無線通信の通信品質や周辺環境の物理的諸要素についても、汎用の計算機と有線のネットワーク上において制御出来る方法によって対処されるべきである。スケーラビリティについては、汎用の計算機を追加することによって容易に規模を拡大出来ることが望ましい。

以上から、StarBED型テストベッドの特性はIoTの実証実験においても有効である。本研究では、StarBED型テストベッドにおけるIoTのネットワークシステムを対象にした大規模実証実験環境構築を提案する。ここでは、StarBED型テストベッドにおけるネットワークシステムの大規模実証実験モデルをIoTのネットワークシステムに対して適用する方法を議論する。

StarBED型テストベッドは汎用PCノードと有線で構築されている。そのため、PCと異なるアーキテクチャで実装されたIoTデバイスの実コードをそのまま実行する事は出来ない。したがって、IoTデバイスを何らかの手法で模倣し、汎用PCノード上で動作させる必要がある。また、無線を用いるネットワークシステムの実証実験を行う場合、何らかの手法で無線通信を模倣し、通信品質の変化を有線のネットワーク上で表現する必要がある。このため、現在のStarBEDにおける実証実験フレームワークにおいては、IoTを対象とした実証実験を行う場合、実験者が環境を一から用意する必要がある。

4.5 IoTデバイスの模倣

アーキテクチャの違いを克服する方法の1つとして、ソフトウェア的に代用する方法が挙げられる。実際のIoTデバイスを用いる以外にも、IoTデバイスの機能をソフトウェア的に代用し、IoTデバイスを模倣することが可能である。本節では、IoTデバイスを抽象化するレベルを概観し、そのレベルでIoTデバイスを模倣した時に観測できるデータの現実に対する迫真性について説明する。

IoTデバイスを抽象化するレベルに応じて、採用すべき手段が異なる。図4.2はIoTデバイスの構成とそれを抽象化するレベルに応じた方式を列挙したものである。IoTデバイスはハードウェアとファームウェアから構成される。ハードウェアはコントローラと電子回路から構成され、ファームウェアはアプリケーション、デバイスに依存しないOSソフトウェア及びデバイスドライバから構成される。以下の各項において、IoTデバイスを抽象化するレベルに応じたそれぞれの方式について説明する。

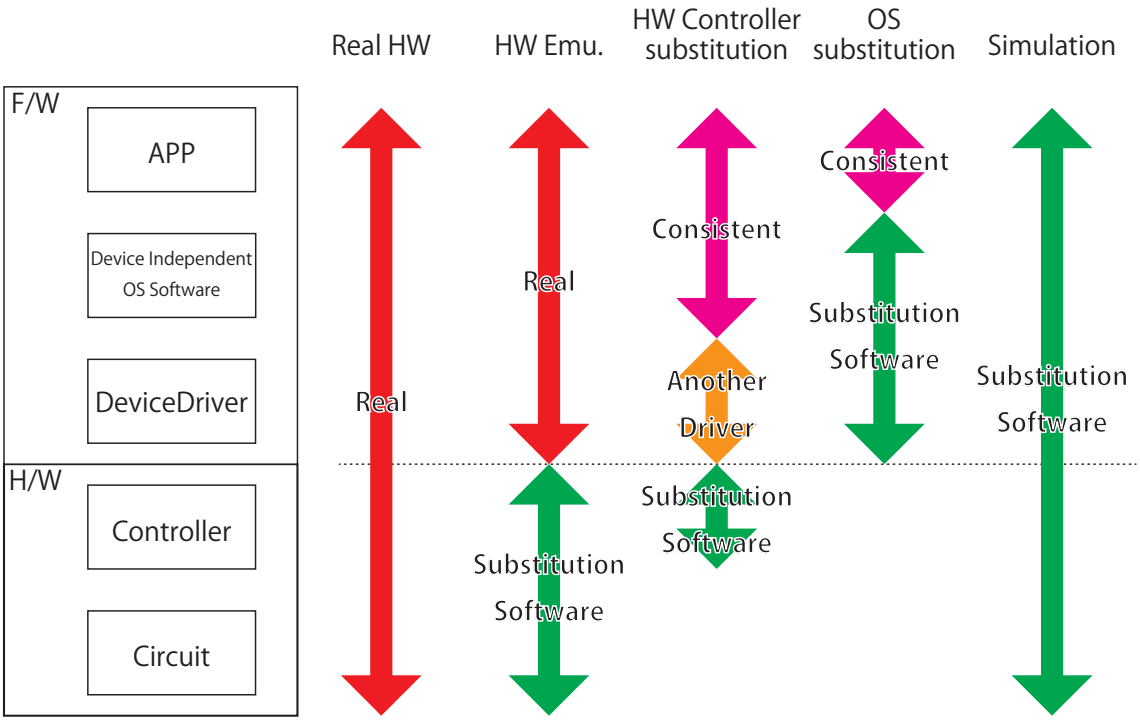

図 4.2: 各方式における実験結果の迫真性の比較

4.5.1 ハードウェアエミュレーション

ハードウェアエミュレーションはハードウェアの全体をソフトウェア的に代用する方法である。OS からは実際のハードウェアと同様に認識され、実際のハードウェアと同じデバイスドライバが使用される。アプリケーションソフトウェアも実際のハードウェア上と同様に動作する。

実際の IoT デバイスと比較した場合、アプリケーションからデバイスドライバまでファームウェアレベルでの一貫性を有している。ハードウェアの動作に関しても考慮する場合、ハードウェアエミュレーションは、他のレベルでの抽象化と比較して実験結果の迫真性が最も高い方法である。

無線のネットワークインタフェースしか持たない IoT デバイスのエミュレータを利用して有線のネットワーク上で通信を行うためには別途対応が必要である。

4.5.2 ハードウェアコントローラのソフトウェア的代用

この方法は、実際の IoT デバイスにおけるハードウェアコントローラに相当する機能をソフトウェア的に代用し、ホストとなる物理ノードのハードウェアを通じて入出力を実現する方法である。OS からは実際のハードウェアと異なるハードウェアとして認識され、デバイスドライバはハードウェアコントローラの代用ソフトウェアとの界面として機能す

る。実際の IoT デバイスにおいて動作するファームウェアと比較した場合、デバイスドライバ以外のソースコードレベルにおいて一貫性を保つことが出来る。

この方法の場合、電子回路の模倣を行わないためハードウェアエミュレーションと比較して実験結果の迫真性は低下する。しかし、電子回路の模倣に必要なリソースを消費しないためスケーラビリティは向上する。

4.5.3 OS のソフトウェア的代用

IoT デバイスのハードウェアに相当する機能をソフトウェア的に代用せず、OS の提供するサービスをソフトウェア的に代用することで、ユーザ空間アプリケーションのみをホストとなる物理ノード上で実行しアプリケーション自体のスケーラビリティを検証する方法も考えられる。

実験結果の迫真性はアプリケーションレベルに留まり、OS およびハードウェアの挙動を確認することは出来ない。

4.5.4 シミュレーション

この方法は IoT デバイスの動作についてモデル化し挙動を確認する。いずれのレベルにおいても実際に用いられる IoT デバイスと一貫性のある模倣を行わない。モデル化の精度に応じて結果の迫真性は変化するが、結果の迫真性は相対的に低い。

検証対象をアプリケーションとした場合、実際の IoT デバイスで用いるアプリケーションと一貫性のある方式はハードウェアエミュレーション、ハードウェアコントローラのソフトウェア的代用、OS のソフトウェア的代用の 3 方式である。

4.6 無線通信の模倣

StarBED 型テストベッドにおけるネットワークは有線で構築されている。他方で、IoT デバイスの利用する通信メディアには無線も含まれる。そのため、有線のネットワーク上で無線通信による通信品質の変化を模倣する手段が必要となる。

4.6.1 QOMET

QOMET[8] は有線ネットワーク上で IEEE 802.11 (Wi-Fi) や IEEE 802.15.4 などで構成される無線ネットワークの通信品質を模倣するネットワークエミュレータである。QOMET はユーザによるシナリオ記述に基いて無線ネットワークの通信品質をシミュレーションする deltaQ と、シミュレーション結果を用いて有線ネットワーク上で無線環境をエミュレー

ションする wireconf から構成される。wireconf による無線通信のエミュレーションはネットワーク層で動作し、IPv4 アドレスを用いて計算機を識別する。このため、IPv4 ヘッダを含まないイーサネットフレームに対して無線通信のエミュレーションを行うことはできない。

4.6.2 Meteor

Meteor[17] はイーサネットフレームに対するフィルタリングを行う無線ネットワークエミュレータである。遅延挿入や帯域制御をデータリンク層で行うことで、ネットワーク層において遅延挿入や帯域制御を行うネットワークエミュレータと違い IPv4 ヘッダを含まないイーサネットフレームへの対応が可能である。これにより、既存のネットワークエミュレータで対応が困難であったプロトコルやアプリケーションソフトウェアの検証を可能としている。

第5章　IoT大規模実証実験フレームワーク：GUAN

4.4節において議論したように、現在のStarBEDにおける実証実験フレームワークではIoTを対象にした実証実験を行う際に、実験者が一から実験環境を作る必要がある。そこで本研究では、StarBED型テストベッドにおけるIoTを対象とした実証実験フレームワークを提案する。本研究において提案するフレームワークを、ユーザが任意に選択したネットワーク技術を一般的に利用可能にするということで、Generic Utilization of Assorted Networking(以下、GUAN) [14] と名づけた。

5.1　仮想IoTデバイス

4.4節において述べたとおり、StarBED型テストベッドにおいて、アーキテクチャの違いからアプリケーションのコードをそのまま実行することが出来ないIoTデバイスが存在する。そのようなIoTデバイスのためのアプリケーションを対象に実験を行う方法として、StarBED型テストベッドの汎用PCノード上で仮想的にIoTデバイスを動作させる方法が必要である。提案フレームワークにおいては、4.5節において議論したように、IoTデバイスを模倣しソフトウェア的に代用することでPC上のプロセスとして動作させる。本論文では、図4.2で示した各方式の中で、ハードウェアエミュレーション、ハードウェアコントローラのソフトウェア的代用、OSのソフトウェア的代用の3方式によりIoTデバイスを模倣したプロセスを仮想IoTデバイス (Virtual IoT Device) と定義し扱う。

仮想IoTデバイスは、ハードウェアエミュレータを用いたり、ハードウェアに依存するソフトウェアの動作に必要なハードウェアが提供する機能をソフトウェア的に代用することで、IoTデバイスで用いられるアプリケーションを動作させIoTデバイスに相当する動作をPC上で実現する方法である。仮想IoTデバイスはPC上においてプロセスとして動作し、プロセスレベルでの多重化により1台のPC上で複数のIoTデバイスに相当する処理を行うことが出来る。

本節では、仮想IoTデバイスを用いるうえで留意する必要がある事項について概観する。

5.1.1 仮想 IoT デバイスに割り当てられる識別子

仮想 IoT デバイスに割り当てられる識別子について説明する。はじめに、仮想 IoT デバイスは PC 上のプロセスとして動作するため、PID が動的に割り当てられる。また、仮想 IoT デバイスの実装に応じて通信インタフェースが識別子を有することもある。これについては後の項で記述する。

一般的に、仮想 IoT デバイスの識別子は仮想 IoT デバイスが生成されてから割り当てられる。また、仮想 IoT デバイスの通信インタフェースも仮想 IoT デバイスのプロセスが起動してから生成される。そのため、仮想 IoT デバイスに割り当てられる識別子を管理する手法が必要がある。ただし、インタフェースの生成と識別子の割り当ては別であり、動的に生成されるインタフェースであっても静的に識別子を割り当てることが出来る場合がある。

5.1.2 仮想 IoT デバイスの生成

大規模実証実験においては規模に応じた数の、異なる識別子を割り当てられた仮想 IoT デバイスを生成する必要がある。異なる識別子を割り当てた仮想 IoT デバイスを生成する方法は、一般的に幾つかの方法が存在する。ここでは、識別子を割り当てるタイミングに応じて取りうる方法を説明する。識別子を割り当てられるタイミングは、アプリケーションのコーディング時、アプリケーションのロード時、プロセスの起動後の 3 つに分けられる。

ハードコーディング 予め割り当てた識別子をファームウェア、またはハードウェアエミュレータを始めとする代用ソフトウェアにハードコーディングしておく方法。識別子の異なる仮想 IoT デバイスの数に応じてファームウェアをコンパイルしたり代用ソフトウェアの設定を用意する必要がある。

ロード時における動的書き換え 仮想 IoT デバイスがロードされる際にバイナリパッチを行い、仮想 IoT デバイスの識別子を動的に書き換える方法。ファームウェアや代用ソフトウェアは 1 つ作ればよく、仮想 IoT デバイスの生成にかかる時間の節約が期待できる。

外部の管理機構から割り当てる DHCP のように仮想 IoT デバイス外部から識別子を割り当てる方法も考えられる。ただし、DHCP は MAC アドレスや EUI-64 などの識別子をハードコーディングしていないと用いることが出来ない。つまり、他の識別子に依存する識別子には用いることが出来ない。

また、仮想 IoT デバイスを複数生成する場合、生成する数に応じて所要時間が線形に増加することが予測される。このため、各実験ノードにおいて分散して生成することが望ましい。

5.1.3 仮想 IoT デバイス間の通信

仮想 IoT デバイスは多くの場合、仮想 IoT デバイスのプロセスが起動してから通信に用いるインタフェースの識別子が決定する。そのため、動的に生成される通信インタフェースと仮想 IoT デバイスの関連付けに関して取りうる方法について考慮が必要である。

また、仮想 IoT デバイスの実装によって異なる通信インタフェースを用いて仮想 IoT デバイス間の通信を実現する必要がある。仮想 IoT デバイスの通信インタフェースの生成は、TAP/TUN をはじめとするホスト OS のカーネル上で取り扱うことが出来るネットワークインタフェースを生成する、仮想 IoT デバイスにあたるプロセス毎にトランスポート層のポート番号を割り当てて通信インタフェースとして扱うなどの方法が存在する。

また、仮想 IoT デバイスが通信に用いるフォーマットについても考慮する必要がある。例えば、MSPSim は CC2420 をはじめとする各種無線チップの動作をエミュレートし、その出力する IEEE 802.15.4 のフレームをカプセル化し TCP のペイロードとして他の MSPSim と通信を行う。Contiki の Minimal-net Platform は TAP インタフェースを通じてイーサネットのフレームを送受信する。このように、仮想 IoT デバイスの通信フォーマットは実装に応じて異なる。そこで、仮想 IoT デバイスの通信インタフェースやフォーマットに応じて通信インタフェースの整合を図り、仮想 IoT デバイス間の通信を中継する機構を実装する必要がある。

5.1.4 仮想 IoT デバイスの制御

実験において仮想 IoT デバイスを用いる際に仮想 IoT デバイスの制御が必要になる。本項では、仮想 IoT デバイスの制御について概観する。

仮想 IoT デバイスの起動 実験の開始に伴って仮想 IoT デバイスを起動させるというシェル上で仮想 IoT デバイスを実行した段階でアプリケーションが起動する、ハードウェアエミュレータの起動コマンドを受けて起動するなど仮想 IoT デバイスを起動するための操作は実装によって変化する。

仮想 IoT デバイスの終了 仮想 IoT デバイスのアプリケーションによるが、一旦電源を投入した後は稼働し続けることを想定したアプリケーションの場合、アプリケーションから仮想 IoT デバイスを終了することは不可能である。ハードウェアエミュレータの場合、停止や終了の操作を行うことが可能であるが大規模分散環境において個々のハードウェアエミュレータを対象に操作を行うことは困難である。

物理的な操作のためのインタフェース ボタンなど物理的な操作を行うインタフェースが付属した IoT デバイスも存在する。ハードウェアエミュレータによっては GUI での制御が可能な実装が存在するが、大規模分散環境において個々の GUI を操作する事は現実的でない。

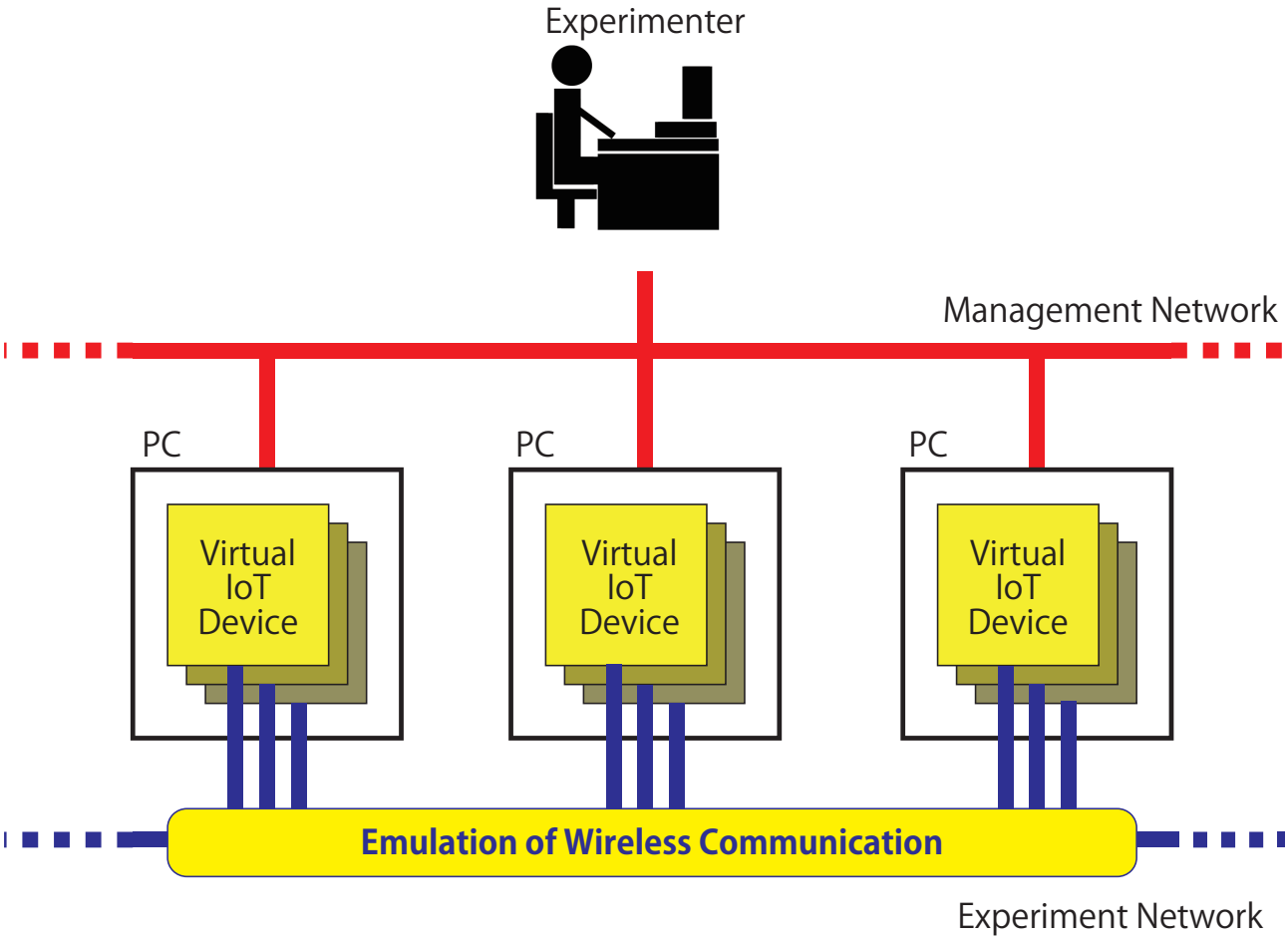

図 5.1: IoT を対象とした実証実験の概形

このように、起動、終了など最も基本的な制御についても実装によって差異がある。そのため、各実装に合わせた操作を逐一行う必要があるが、大規模分散環境において個々の仮想 IoT デバイスを操作することは困難である。そこで、仮想 IoT デバイスの実装に関わらず統一的な制御が可能となる機構が必要である。

5.2 GUAN の構成

図 5.1 は、StarBED 型テストベッドにおける IoT を対象とした実証実験の概形を表している。複数の PC ノードを利用し、それらの PC は管理ネットワークと実験ネットワーク双方に接続されている。実験ノード上に仮想 IoT デバイスを生成し、実験ネットワークを通じて仮想 IoT デバイス間の通信をエミュレーションする。仮想 IoT デバイス間の通信について、無線通信を行うことを想定したネットワークの場合は通信品質の変化をエミュレー

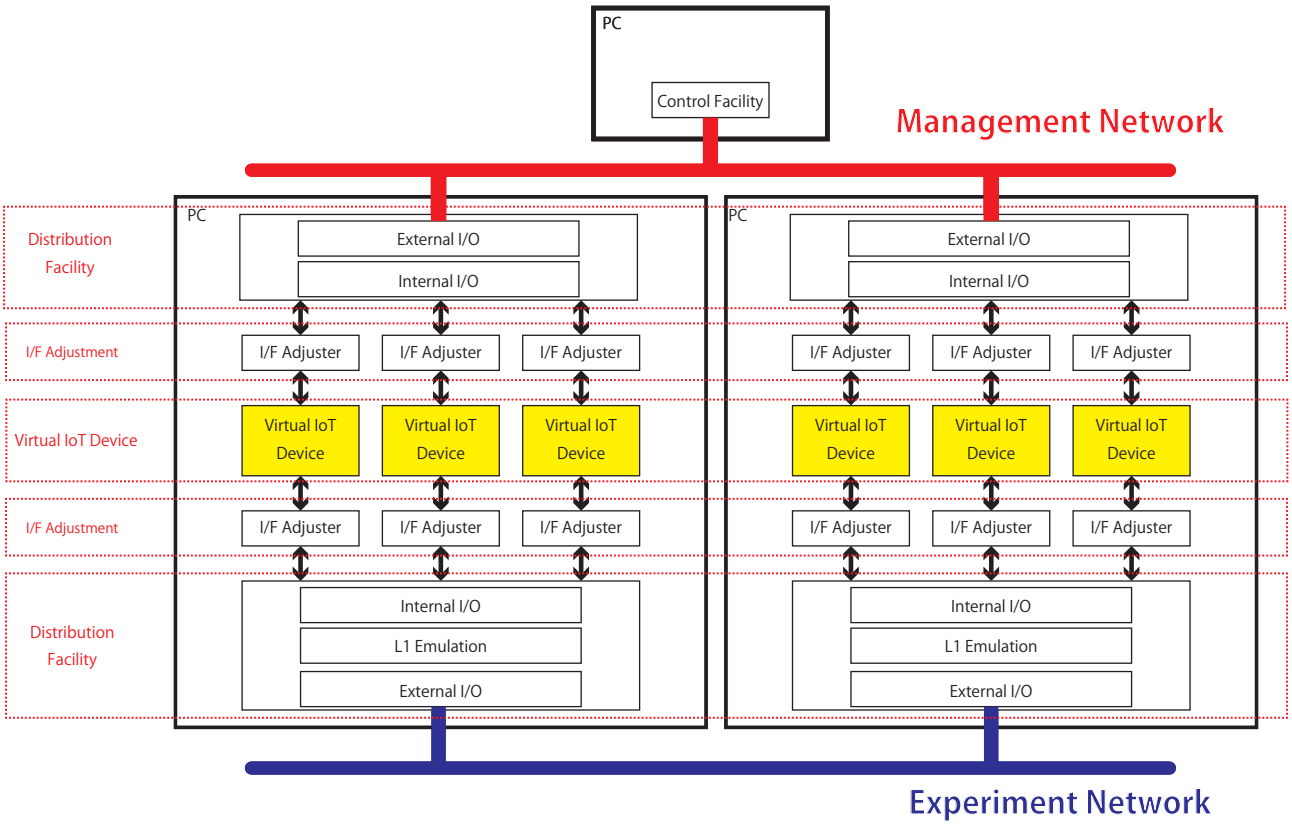

図 5.2: GUAN のアーキテクチャ

ション出来ることが望ましい。また、仮想 IoT デバイスの制御は管理ネットワークを通じて行い、実験で発生する通信と分離可能にすることで、より迫真性の高い実験結果が得られることが期待される。

図 5.2 は GUAN のアーキテクチャを示している。GUAN は仮想 IoT デバイスレイヤを中心に管理ネットワーク側と実験ネットワーク側の双方にインタフェース調整レイヤ、配送機構レイヤの 2 つのレイヤを配置した 3 つのレイヤから構成される。

5.2.1 仮想 IoT デバイスレイヤ

仮想 IoT デバイス (Virtual IoT Device) レイヤでは、仮想 IoT デバイスそのものに極力加工を施すことなく仮想 IoT デバイスを利用可能とするため、次に説明するインタフェース調整レイヤで扱うインタフェースを抽象化する。このレイヤで抽象化されるべきインタフェースとして、起動、終了、物理的な操作などのための制御インタフェース、通信インタフェースが存在する。

GUAN の提案段階 [14] ではこのレイヤをハードウェアエミュレーションレイヤとしていたがハードウェアエミュレータ以外の仮想 IoT デバイスを利用することを考慮し、仮想 IoT デバイスレイヤと改称した。

5.2.2　インタフェース調整レイヤ

インタフェース調整 (I/F Adjustment) レイヤでは、仮想 IoT デバイスの対する入出力および制御のインタフェース整合を実現する。多くの種類の仮想 IoT デバイスを一般的に利用可能にするため、仮想 IoT デバイス毎に割り当てられる識別子や通信の入出力方法など、実装毎に異なる部分の抽象・汎化を実現する。

5.2.3　配送機構レイヤ

配送機構 (Distribution Facility) レイヤは仮想 IoT デバイスが生成する通信や制御メッセージの配送に係る一連の処理を行う。このレイヤは以下の 3 つのサブレイヤから構成される。

内部 I/O サブレイヤ 内部 I/O (Internal I/O) サブレイヤは仮想 IoT デバイスに対する制御やネットワーク通信の入出力を取り扱う。このサブレイヤは、仮想 IoT デバイスから出力されインタフェース調整レイヤの処理を受けた通信データを宛先に向けて送信する、外部 I/O、L1 エミュレーションサブレイヤの処理を受けた通信を宛先の仮想 IoT デバイスに付随するインタフェース調整レイヤに入力するなどの処理を行う。

外部 I/O サブレイヤ 外部 I/O (External I/O) サブレイヤは PC が接続されているネットワークへの入出力を取り扱う。管理ネットワークにおいては、制御機構との通信を実現する。実験ネットワークにおいては、他の PC との間で仮想 IoT デバイスの通信を中継する。これによって、異なる PC 上で動作する仮想 IoT デバイス間の通信を実現する。

L1 エミュレーションサブレイヤ L1 エミュレーション (L1 Emulation) サブレイヤは、実験ネットワーク側の配送機構レイヤにのみ存在する。GUAN の構成する実験ネットワーク上で、仮想 IoT デバイス間や他のサーバと仮想 IoT デバイスの通信などに無線通信の模倣を適用する。

第6章　GUANの実装例

6.1　利用する仮想IoTデバイス

仮想IoTデバイスとして、Contiki-OSのMinimal-net Platformに加工を施したプログラムを用いる。図6.1はMinimal-net Platformの構成を示している。Minimal-net PlatformはContiki-OSの利用形態の一種であり、主にアプリケーションの開発やデバッグにおいて用いられる。Minimal-net Platformの構成は、Contiki-OS向けに実装したアプリケーション、ハードウェアに依存しないOSのソフトウェアとドライバからなるContiki-OS、ホストOSを経由して通信を行うための機構をはじめとする適合をはかるためのソフトウェアで構成される。ホストOS側から見ると、Minimal-net Platformは1つのプロセスとして動作する。

デフォルトのMinimal-net PlatformはTAPインタフェースの識別子や各仮想IoTデバイスに割り当てられるIPアドレス(IPv4およびIPv6のいずれか)、MACアドレスを自由に設定することが出来ない。そのため、Minimal-net Platformによる仮想IoTデバイスを複数利用するための加工が必要である。

本研究では、先に上げたTAPインタフェースの識別子への対応やIPアドレス、MACアドレスの書き換えを行うための加工を施し、コンパイル前の段階で動的に識別子を変更するシェルスクリプトを実装した。

6.2　制御機構

仮想IoTデバイスの制御を行うための実験ノード制御にはcsshX[1]を用いた。csshXは複数の対象にssh接続をし、統一した操作を行うことが出来るほか、個別の端末を操作し各接続先に異なる操作を行うことも出来る。ユーザ端末としてMacを利用する場合、csshXを用いることで、各ノードに統一的な制御を行うことが容易になる。

6.3　仮想IoTデバイスの制御

本節では、管理ネットワーク側のインタフェース調整レイヤに相当する仮想IoTデバイスの制御に関する実装を説明する。仮想IoTデバイスの制御の中でも、生成、起動、終了

[1]Cluster SSH tool using Mac OS X Terminal.app

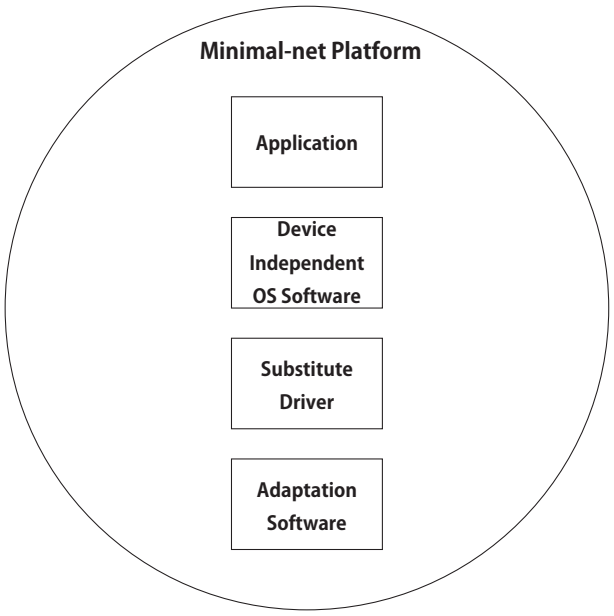

図 6.1: Minimal-net Platform による仮想 IoT デバイスの概要

および標準出力の処理について説明する。

6.3.1 仮想 IoT デバイスの生成

前節で述べた、動的に識別子を変更するシェルスクリプトを実行し、その後、コンパイルを実行する。割り当てる識別子を変更しつつこの処理を任意の回数繰り返すスクリプトを実装し、それを用いた。

バイナリの名前については、生成時に一意に識別出来る名前を付けておくことで、process status (ps) からの参照を容易にする。また、仮想 IoT の終了の項で述べるが、プロセスの終了においても重要な役割を持つ。本実装ではプレフィックス、ID、サフィックスからなる命名規則を作成し、生成されたバイナリの識別子として扱う。プレフィックス、ID、サフィックスについては次のように定義した。

プレフィックス
プラットフォームの略号を用いる。本実装で用いるプラットフォームは Minimal-net Platform なので mn と略する。

ID
仮想 IoT 毎に一意に設定する。また、MAC アドレスおよび IPv6 アドレスの設定にも用いる。

サフィックス バイナリに付与される拡張子をそのまま使用する。ここでは.minimal-net となる。

6.3.2 仮想 IoT デバイスの起動

Minimal-net Platform による仮想 IoT デバイスはコマンドラインから実行する。実行対象の仮想 IoT デバイス毎に、標準出力をリダイレクトしログファイルを出力させ、また、バックグラウンドで実行する必要がある。以上の機能を持つシェルスクリプトを実装した。

6.3.3 仮想 IoT デバイスの終了

Minimal-net Platform による仮想 IoT デバイスには操作のためのインタフェースが存在しない。このため、コマンドラインからプロセスを終了するために SIGINT を送ることで終了する。本実装においては、バックグラウンドで多数の仮想 IoT デバイスを動作させているため、pkill を用いて該当する名前のプロセスにシグナルを送り終了する。

6.3.4 仮想 IoT デバイスからの標準出力および標準エラー出力の処理

仮想 IoT デバイスの標準出力をログとして収集する。実験における検証対象は各仮想 IoT デバイス上で動作するアプリケーションソフトウェアである。デバッグをはじめ後にログの参照を必要とした際に切り分けを容易にするため、各仮想 IoT デバイス毎に異なるログファイルを出力する。

6.4 仮想 IoT デバイス間の通信

本節は GUAN のアーキテクチャにおける実験ネットワーク側の配送機構レイヤに相当する仮想 IoT デバイスの通信に関する実装について説明する。

Minimal-net Platform による仮想 IoT デバイスは TAP を生成しホスト OS との通信に用いる。この TAP インタフェースを利用して、仮想 IoT デバイス間の通信や他のサーバとの通信に利用する方法を考察する。

同一実験ノード上の仮想 IoT デバイス間通信

フラットな L2 ブリッジインタフェースを作成し、各仮想 IoT デバイスの生成した TAP インタフェースをブリッジインターフェースに追加する。これにより、各仮想 IoT デバイスは同一の L2 上に存在し、通信を行うことが可能となる。

L2 の分割 複数のブリッジインタフェースを作成し、各仮想 IoT デバイスをそれぞれ分割したいブリッジインタフェースに追加する。

異なる実験ノード上の仮想 IoT デバイス間通信

フラットな L2 実験スイッチに接続されているイーサネットインタフェースをブリッジに加えることで、実験ネットワークと同一のデータリンクに仮想 IoT デバイスを配置することが可能である。

L2 の分割 同一ノード上の仮想 IoT デバイス間通信において用いたブリッジインタフェースを複数作成する方法に VLAN を組み合わせて使う方法が考えられる。

第7章　GUANによる大規模実証実験例

7.1　実験概要

本節では、GUANの概念実証実験として、複数のPCを用いてそれらの上で分散して実行される仮想IoTデバイスとサーバ間の通信を実現する。図7.1は、実験の概形を表している。以下、データ収集を行うプログラムが動作する実験ノードをサーバと呼び、仮想IoTデバイスが動作する実験ノードをクライアントと呼ぶ。

サーバ上では、クライアント上で動作する仮想IoTデバイスから送信されたデータの収集を行うプログラムを動作させる。クライアント上では、Minimal-net Platformに加工を加えた仮想IoTデバイスを利用し、定期的にサーバにデータを送信する仮想IoTデバイスを複数動作させる。実験には1台のサーバと8台のクライアントを用いた。動作させる仮想IoTデバイスの数量を変更することで得られる結果の差異については後述する。

7.2　実験環境の構成

実験施設はStarBEDのグループHを借用した。グループHを構成する汎用PCノードのスペックは表7.1を参照されたい。

サーバにおいて、ブリッジインタフェースを作成し、各仮想IoTデバイスのデータ送信先として指定したIPアドレスを割り当てる。このブリッジインタフェースに、実験スイッチに接続されているイーサネットインタフェースを追加する。

各クライアントでは、各仮想IoTデバイスが起動した際に生成するTAPインタフェースを各実験ノードのブリッジインタフェースに追加する。さらに、実験スイッチに接続されているイーサネットインタフェースも同様にブリッジインタフェースに追加することで、同一のデータリンク上に多数の仮想IoTデバイスが存在する実験ネットワークが構成される。

7.3　実験ノードの構築

グループHを利用して均質なノード群に展開するため、基本となる構成のノードを1台予め作成する。OSをインストールし、実験に関連する環境を構築したディスクのイメージを複数の実験ノードに複製することで均質な実験ノードを構築する。

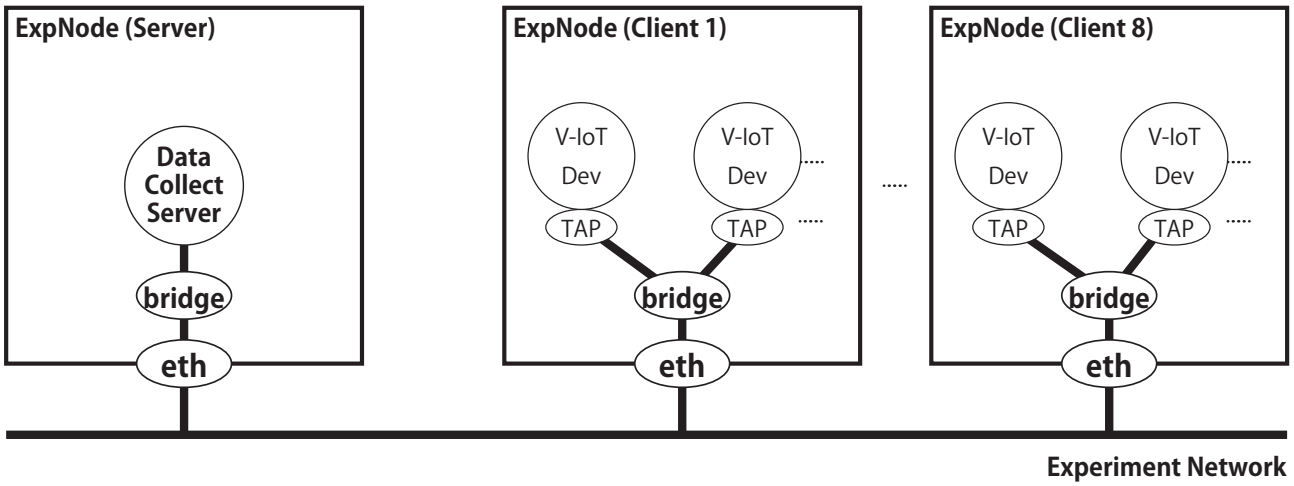

図 7.1: 実験の概形

表 7.1: グループ H を構成する汎用 PC ノードのスペック

Model	HP DL320 G5
Chipset	Intel 3210 (1333 MHz FSB)
CPU	Intel Xeon X3350 (2.66 GHz / 12 MB L2 Cache / 1333 MHz FSB / 4 Core) x 1
Memory	2 GB unbuffered DIMM (DDR2-800 / PC2-6400 / ECC / SDRAM) x4
HDD	160 GB 3.5 in HDD (SATA / 7.2K RPM) x 1
Optical Drive	SATA DVD-ROM drive
NIC (on-board)	double Gigabit Ethernet HP NC326i Integrated Dual Port PCI Express Gigabit Server (for management x 1 / for IPMI x 1)
NIC (extend)	double Gigabit Ethernet HP NC360T PCI Express Dual Port Gigabit Server Adapter (for experiment x 2)

blanketsh blanketsh は StarBED が提供する SpringOS の機能を利用するためのユーザインタフェースである。本実験においては、getdisk および distdisk によるディスクイメージの作成、配布機能や、VLAN 設定、実験ノードの電源投入などに用いた。

7.3.1 OS のインストール

StarBED においてインストールメディアを用いて OS のインストールを行う場合、一般的な PC に OS をインストールする方法と特に違いはない。本項では、StarBED の実験支援機構を利用して実験環境を複製する際に留意すべき事項を挙げる。

ボリュームの大きさ ディスクイメージを作成して他の実験ノードに配布するため、ボリュームの大きさを必要最小限に抑制することで、複製に必要な時間を短縮することが出来る。

udev Linux はカーネルバージョン 2.6 以降で udev による動的デバイス管理が実装されている。通常ならばカーネルによって発見順に識別子が割り当てられるため、システムの再起動によって識別子が変化することもあったが、これにより、再起動後も同じ識別子が期待通りに割り当てられる。便利な機能であるが、ディスクイメージを作成して他の PC に複製するような用途においては、ネットワークインタフェースの識別子の割り当てに影響が出る。例えば、StarBED 型テストベッドにおいて、実験に関連する環境を構築する際に、ネットワークを利用してパッケージを取得することは頻繁に行われる。ネットワークインタフェースに識別子が割り当てられ、/etc/udev/rules.d/70-persistent-net.rules に MAC アドレスとの対応関係が記録される。この状態のディスクイメージを保存し、他のノードに複製した場合、ネットワークインタフェースに割り当てられた識別子と MAC アドレスの組と異なる MAC アドレスを有するネットワークインタフェースが順次発見されるため、先に記録された識別子の次の識別子から割り当てられることになる。これは、均質な実験ノードを用いて実験を行う StarBED 型テストベッドにおいて識別子を予見することが困難になる。

ディスクイメージの作成前に/etc/udev/rules.d/70-persistent-net.rules が存在する場合はこれを消しておく。こうすることで、ディスクイメージを配布し複製した実験ノードにおいてもネットワークインタフェースの識別子は最初から割り当てられる。

7.3.2 ディスクイメージの作成と配布

実験に必要な最小限のパッケージをインストールしたディスクイメージを作成する。blanketsh を利用し、このディスクイメージを保存し、配布を行う。ディスクイメージを配布し複製した実験ノードはホスト名が複製元と同じになっている。実験において仮想 IoT デバイスに割り当てる識別子や実験のログを取得する際に、ホスト名と対応した識別子を利用する。そのため、この段階で hostname の変更を行っておく。また、複数の実験ノー

ドからログを取得した場合、ログの比較において時刻が重要である。そのため、実験ノードの時刻を同期しておく必要がある。時刻の同期手法には Network Time Protocol (NTP) がある。本実験においても、実験ノードの時刻同期は StarBED が提供している NTP サーバを利用した。

7.4 評価実験

定期的に所定のサーバへデータを送信するセンサの基本的な動作を模擬したアプリケーションを作成し、仮想 IoT デバイスとして実行する。並列分散したクライアントから送信されたデータがサーバで正しく受信出来ているかを検証した。

サーバの挙動は以下の通りである。

- 所定のポートで仮想 IoT デバイスからの UDP データグラムを待ち受ける。
- 仮想 IoT デバイスからのデータグラムを受信したら、受信時刻を挿入し、仮想 IoT デバイスに返送する。

仮想 IoT デバイスで実行するアプリケーションの大まかな挙動は以下の通りである。

1. バイナリが実行されると TAP インタフェースを生成する (Minimal-net Platform 共通の挙動)。
2. 15 秒待機。
3. サーバに対してシーケンス番号の入ったメッセージを UDP データグラムのペイロードとして送信。
4. 以下、10 秒毎にシーケンス番号を増加させて同様にメッセージ送信を繰り返す。

＊ サーバからの応答があった場合、それを表示する。

各クライアント上で生成する仮想 IoT デバイスの数を変更し、仮想 IoT デバイスの生成にかかる時間や実験ネットワークにおける通信について観測する。

7.4.1 仮想 IoT デバイスの生成

図 7.2 は、GroupH の実験ノードにおいて本実験で用いる仮想 IoT デバイスを生成するためにかかった時間を示している。生成する仮想 IoT デバイスの増加に伴って、所要時間も線形で増加する。

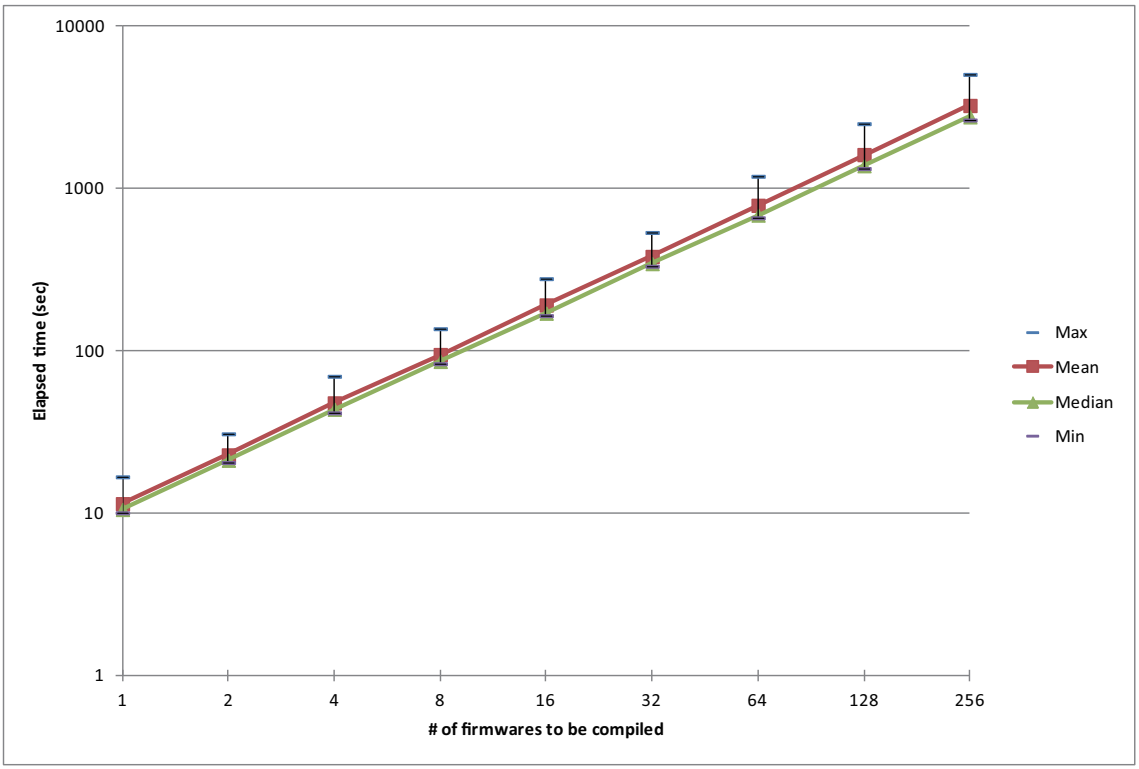

図 7.2: ファームウェアの生成にかかる時間

7.4.2　実験ネットワークへの接続

表 7.2 は各クライアント上で動作する仮想 IoT デバイスからサーバに送信されるメッセージ数の予測値とシーケンス毎の実測値である。各クライアントにおいて 128 台の仮想 IoT デバイスを生成した実験において、計 1024 台の仮想 IoT デバイスから一斉に送信されたメッセージが観測されるまでに 2 回、予測値を下回った。これは、TAP インタフェースをブリッジインタフェースに追加する処理が終了する前に仮想 IoT デバイス起動時における 15 秒の待機が終わり、仮想 IoT デバイスのメッセージ送信が行われたためである。

TAP インタフェースはアプリケーションが起動した後に作成されるため、仮想 IoT デバイス数の増加に比例して TAP インタフェースの操作に係る時間が増加し、コンテキストスイッチの関係で生成した全ノードへの疎通が確認出来るまでに必要な時間が急激に増加する傾向があると考えられる。

7.4.3　データ収集サーバのエラー終了

各実験ノードで仮想 IoT デバイスを 256 個動作させた場合、データ収集サーバがエラー終了したため計測不能となった。そのため図 7.3 のグラフ上にデータは反映していない。

表 7.2: 受信メッセージ数の予測値と実測値

# of V-IoT Dev.	Est. #	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
1	8	8	8	8	8	8
2	16	16	16	16	16	16
4	32	32	32	32	32	32
8	64	64	64	64	64	64
16	128	128	128	128	128	128
32	256	256	256	256	256	256
64	512	512	512	512	512	512
128	1024	530	976	1024	1024	1024
256	2048	260	537	805	1078	N/D

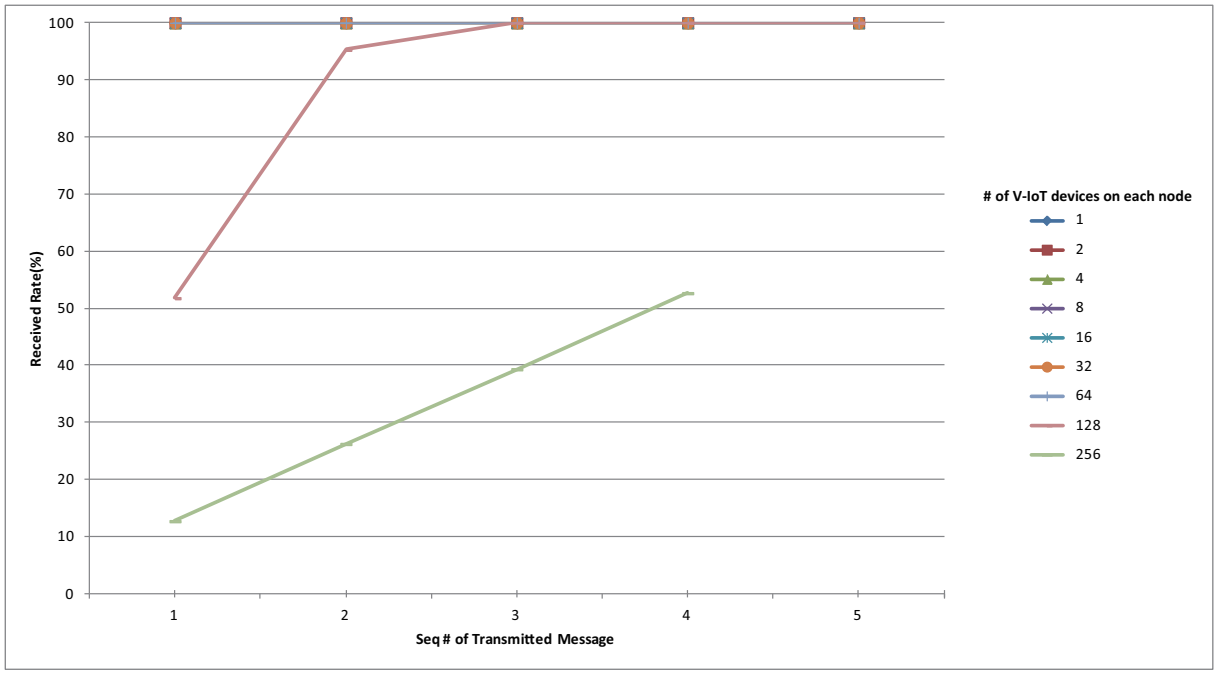

図 7.3: 受信メッセージ数の割合

データ収集サーバのエラー終了の原因は、各仮想 IoT デバイスからのデータを受信した後、受信時刻を挿入して応答を返す処理において発生するソケットエラーである。

7.4.4 追加実験

データ収集サーバの実装を変更して追加実験を行った。データ収集サーバに関して変更した点は、「仮想 IoT デバイスからのデータグラムを受信したら、受信時刻を挿入し、仮想 IoT デバイスに返送する。」処理の削除である。

仮想 IoT デバイスを一斉に起動し、TAP インタフェースをブリッジインタフェースに追加した後に、300 秒待機し、仮想 IoT デバイスの終了を行った。

サーバにおいて計測したメッセージ数の実測値と予測値、受信メッセージ数の割合は表 7.4.4 に示したとおりであり、図 7.4 はこの結果における受信メッセージ数の割合をグラフ化したものである。

この追加実験において、各仮想 IoT デバイスが作成したクライアント毎に 256 個の TAP インタフェースをブリッジインタフェースに追加するために、各ノードが要した時間は平均で 78.38 秒であり、最大で 83.7 秒であった。つまり、開始から 85 秒経過に相当するシーケンス番号 8 のメッセージが送信される段階においてはほぼすべての仮想 IoT デバイスが同一データリンクに接続していると推定できる。しかし、データ収集サーバにおいて受信出来たのはシーケンス番号 8 の 96.63%が最大である。

サーバ、クライアントの各ブリッジインタフェースにおいてパケットキャプチャを行い、

各ブリッジインタフェースにおいて観測された UDP データグラムを分析すると、クライアントのブリッジインタフェースにおいて各シーケンス毎に予測される数の UDP データグラムが観測出来なかった。シーケンス番号 8 と 32 について、各ブリッジインタフェースでキャプチャされたデータグラム数は表 7.4 のとおりである。各シーケンス毎に 256 個の仮想 IoT デバイスがデータグラムを送信しているが、ブリッジインタフェースにおいて 256 個のデータグラムをすべてキャプチャ出来たのはクライアント 2 におけるシーケンス番号 32 のデータグラムのみである。

また、キャプチャ出来たデータグラム数が最も少なかったのはシーケンス番号 19 の 62.21%である。この時にキャプチャ出来たデータグラム数が最も少なかったのはクライアント 8 における 137 個である。

以上で示した仮想 IoT デバイスを複数生成し、同一データリンク上に存在するデータ収集サーバにデータを送信する実験を行った。この実験によって、仮想 IoT デバイスが作成する TAP インタフェースをブリッジインタフェースに追加し、ブリッジインタフェースを通じて仮想 IoT デバイスの通信を取り扱う場合に、ブリッジインタフェースがすべてのパケットを処理出来なくなる現象が発生することが判明した。

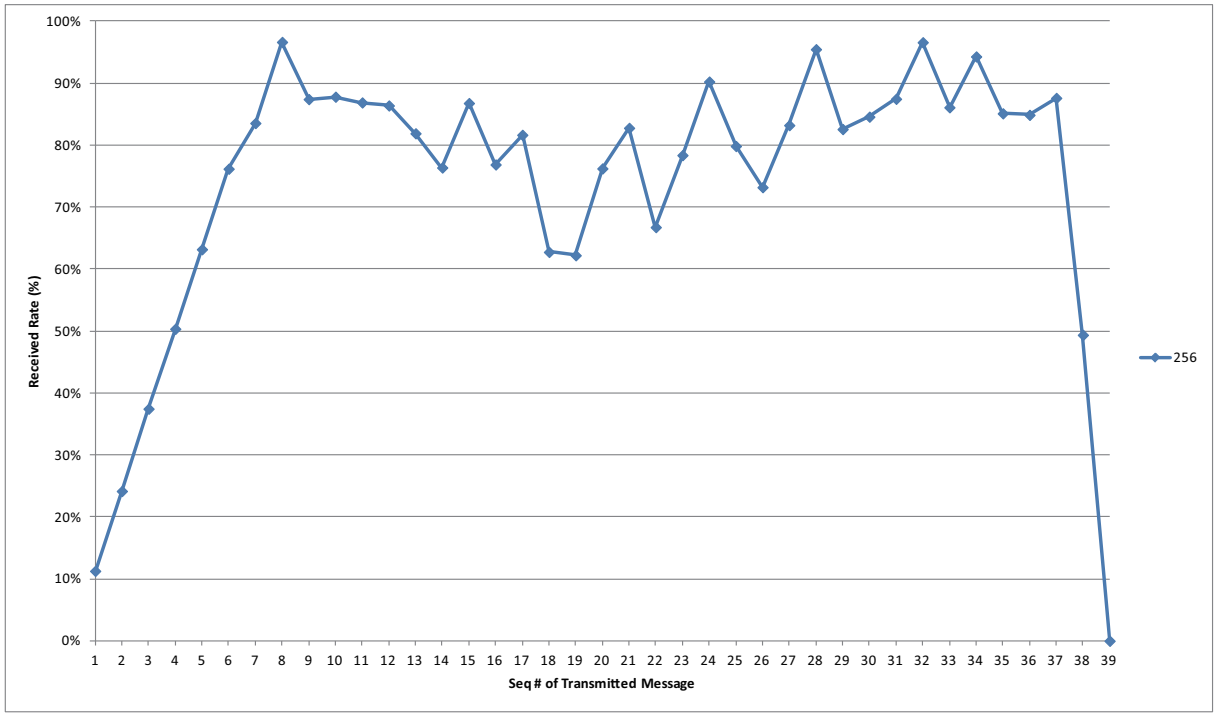

図 7.4: 追加実験における受信メッセージ数の割合

表 7.3: 追加実験における受信メッセージ数の予測値と実測値

Seq #	Measured value	Estimated value	Received Rate
Seq 1	231	2048	11.28%
Seq 2	495	2048	24.17%
Seq 3	767	2048	37.45%
Seq 4	1031	2048	50.34%
Seq 5	1294	2048	63.18%
Seq 6	1560	2048	76.17%
Seq 7	1711	2048	83.54%
Seq 8	1979	2048	96.63%
Seq 9	1790	2048	87.40%
Seq 10	1798	2048	87.79%
Seq 11	1779	2048	86.87%
Seq 12	1769	2048	86.38%
Seq 13	1677	2048	81.88%
Seq 14	1564	2048	76.37%
Seq 15	1777	2048	86.77%
Seq 16	1574	2048	76.86%
Seq 17	1672	2048	81.64%
Seq 18	1285	2048	62.74%
Seq 19	1274	2048	62.21%
Seq 20	1561	2048	76.22%
Seq 21	1695	2048	82.76%
Seq 22	1367	2048	66.75%
Seq 23	1605	2048	78.37%
Seq 24	1848	2048	90.23%
Seq 25	1635	2048	79.83%
Seq 26	1499	2048	73.19%
Seq 27	1704	2048	83.20%
Seq 28	1955	2048	95.46%
Seq 29	1691	2048	82.57%
Seq 30	1732	2048	84.57%
Seq 31	1791	2048	87.45%
Seq 32	1978	2048	96.58%
Seq 33	1763	2048	86.08%
Seq 34	1931	2048	94.29%

Seq #	Measured value	Estimated value	Received Rate
Seq 35	1743	2048	85.11%
Seq 36	1738	2048	84.86%
Seq 37	1794	2048	87.60%
Seq 38	1011	2048	49.37%
Seq 39	0	2048	0.00%

表 7.4: 各ブリッジインタフェースでキャプチャされたメッセージ数 (抜粋)

Seq #	Cli. 1	Cli. 2	Cli. 3	Cli. 4	Cli. 5	Cli. 6	Cli. 7	Cli. 8	Total
seq 8	238	245	247	254	251	248	251	245	1979
seq 19	163	169	184	150	155	162	154	137	1274
seq 32	255	256	232	249	255	243	246	242	1978

第8章　評価と議論

8.1　評価

本研究では、IoTを対象とした実証実験において複数の計算機を用いる実証実験フレームワークGUANを提案した。本節では、GUANの評価を行う。本節では評価にあたって、MSPSim/COOJAを比較対象として用いる。

8.1.1　ネットワーク構成に関する評価

StarBED型テストベッドの実験ネットワークは予め申請し、割り当てられた範囲のVLAN IDを用いて分割することが可能である。そのため、StarBED型テストベッドを用いて実験環境を構築する場合、実験者が自由にネットワークを構築することが出来る。

MSPSim/COOJAは座標空間上に仮想IoTデバイスのノードを配置するため、完全に分離したネットワークを構成することが出来ない。また、MSPSim/COOJAは無線通信によるネットワークをシミュレーションするのみであり、有線で構築されたネットワークを対象に実験を行うことが不可能である。

GUANはStarBED型テストベッドを利用し、VLAN IDによって実験ネットワークを分割することが可能である。このため、ネットワーク構成においてより自由なネットワークが構築可能である。以上がネットワーク構成に関する評価である。

8.1.2　アーキテクチャの混在に関する評価

汎用PCノードと有線のネットワークで構成されるStarBED型テストベッドにおいて、PC以外のアーキテクチャを採用したIoTデバイスのアプリケーションをそのまま実行することは出来ない。そこで、本研究では仮想IoTデバイスを用いることで、アプリケーションの検証を可能とした。

本研究における実験のための実装では、ハードウェアコントローラのソフトウェア的代用によってアプリケーションをPC上で実行した。OSソフトウェアのレベルで一貫性があり、計算機アーキテクチャの違いはコンパイラによって吸収される。

GUANの設計上、仮想IoTデバイスとしてハードウェアエミュレータを用いれば、異なるアーキテクチャのIoTデバイスを模倣することが出来る。

MSPSim/COOJA においては、MSPSim によるハードウェアエミュレーションと Java Native Interface によってハードウェアエミュレーションを伴わない仮想 IoT デバイスを用いて実験を行うことが出来る。

異なる計算機アーキテクチャの IoT デバイスに関しては、GUAN も MSPSim/COOJA も同じ抽象度の仮想 IoT デバイスを用いることが出来る。しかし、MSPSim/COOJA においては PC をシミュレーション上のノードとして配置することが出来ない。他方、GUAN は実験ノードとして PC を用いることも可能である。この点で GUAN に優位性がある。以上が、アーキテクチャの混在に関する評価である。

8.1.3 通信メディアの混在に関する評価

StarBED 型テストベッドは有線のネットワークを用いて構築されている。そのため、無線通信プロトコルを利用する仮想 IoT デバイスを用いた実験環境を構築する場合、通信品質の変化の模倣や、無線通信プロトコルを取り扱う機構が必要となる。

GUAN のアーキテクチャの設計上、各実験ノードの配送機構において無線通信のエミュレーションを行うことを想定している。通信品質の変化は Meteor を用いて模倣することで実現可能である。また、設計上、インタフェース調整レイヤにおいて、通信に用いるインタフェースを調整可能にすることを想定している。このため、仮想 IoT デバイスが IEEE 802.15.4 で通信を行う場合、インタフェースの調整を行ったうえで配送機構を通じて他の仮想 IoT デバイスと通信を行うことは枠組みの上において可能である。

MSPSim/COOJA は MSPSim において CC2420 をはじめとする IEEE 802.15.4 の無線チップをエミュレーションする。このため、ハードウェアドライバのレベルで IEEE 802.15.4 を扱うことが可能であり、6LoWPAN を扱うことも可能である。しかし、先述のとおり、COOJA において行えるネットワークシミュレーションは無線ネットワークのみである。このため、通信メディアの混在には対応出来ない。

以上が、通信メディアの混在に関する評価である。GUAN は設計上通信メディアの混在を想定しており、この要件を満足することができる。

8.1.4 物理的な諸要素の制御に関する評価

IoT デバイスではボタン操作、センサへの入力、LED の点灯、アクチュエータの制御など物理的な諸要素に関連した入出力を伴うアプリケーションも扱われる。これらの入出力を制御する方法に関して、GUAN の設計においては、管理ネットワーク側のインタフェース調整レイヤにおいて扱うことを想定している。GUAN の設計においては、仮想 IoT デバイスに対する物理的な制御命令を入出力するインタフェースを、管理ネットワークから制御可能にするインタフェースの調整である。そのため、物理的な諸要素の制御については、仮想 IoT デバイスの実装に依存する。

MSPSim/COOJA は、IoT デバイスのハードウェアエミュレーションも統合されているため、LED の点灯を始めとする物理的な諸要素の制御についてもシミュレーションされている。

物理的な諸要素の制御に関して整理すると、GUAN は制御インタフェースの調整を行うことのみを設計している。そのため、物理的な諸要素を GUAN は直接制御しない。MSPSim/COOJA は物理的な諸要素に関するハードウェアのエミュレーションも含めたシミュレーションを提供する。

8.1.5 スケーラビリティに関する評価

実時間で実験可能な仮想 IoT デバイスを用いた実証実験の規模にスケーラビリティを有することも、IoT の大規模実証実験に関する要件である。GUAN に基いて構築した実験環境では、計算機の追加によって実験規模の拡張を容易に行えるようになった。

MSPSim/COOJA は、1 台の PC 上で実行するネットワークシミュレータである。MSPSim/COOJA はハードウェアエミュレータの動作も含めてイベントを管理し、シミュレーションを行うため、実時間で動作する仮想 IoT デバイス数は 20 から 30 程度である。また、MSPSim/COOJA はシミュレーション外の計算機やネットワークと通信する手段が限定的であり、複数の計算機を用いた並列分散環境において複数のシミュレータを協調動作させることで実験規模を拡大することは困難である。

GUAN は、実験においては 1000 ノードを超える仮想 IoT デバイスを実時間で動作させることが可能であることを確認した。また、実験ノードを増やすことによって、実時間で動作する仮想 IoT デバイス数を増加させることは容易に可能である。以上から、スケーラビリティに関する要件を GUAN は満足していると評価した。

8.2 議論

8.2.1 実験の実時間性

本研究において行った実験では、Minimal-net Platform に加工を施し仮想 IoT デバイスを実現した。このため、実験で用いた仮想 IoT デバイスにおいてはマイクロコントローラや無線チップなどハードウェアのエミュレーションは行わず、OS を含めて実験ノード上の 1 プロセスとして動作する。実験ノードのインターバルタイマによってアプリケーションソフトウェアが動作するため、ハードウェアエミュレーションと比較して実時間での動作が容易である。

例えば、COOJA の場合シミュレーション中で仮想 IoT デバイスの動作すべてを管理することで、実験ノードの性能に依存するが、実時間より早く実験を行うことも可能である。COOJA ではマイクロコントローラのクロック同期のために tick time を最小単位として実験全体のイベントを管理している。そのため、多数の仮想 IoT デバイスを実験ネット

ワーク上に生成すると、イベント管理のためにより多くのリソースが必要となり、実時間でシミュレーションを動作させることができなくなる。

IoT はセンサネットワークのような単一のアプリケーションで終始するネットワークシステムとならない。ネットワーク間の相互接続を考慮する場合、実験者は双方のネットワークの動作を同期させる必要がある。これは、ネットワークの数が増えることで時間同期が複雑となるほか、ネットワークの相互接続に利用するネットワーク機器が単位時間に処理することが出来る通信に限界がある。このため、複数の PC を利用し構成した分散環境で実験を行う場合、実時間と差異のある速度、特に、実時間より速い時間進行で実証実験を行うことは困難である。

8.3 今後の展望

8.3.1 仮想 IoT デバイスの生成手法

本研究において行った実験では、ファームウェアのコンパイル前にハードコーディングする方式で仮想 IoT デバイスのバイナリを生成した。バイナリの生成は実験の規模に比例して所要時間が増大するため、より大規模な実験を行う際にバイナリの生成に大きな時間をかけることは実験の主目的でなく、望ましくない。このため、仮想 IoT デバイスの生成方法についてより効率的な手段が求められる。

8.3.2 仮想 IoT デバイスの制御性について

本研究において行った実験では、仮想 IoT デバイスの起動後、TAP インタフェースが生成されブリッジインタフェースに追加する処理を行う。この処理は、実験環境の初期化に相当し、アプリケーションの自体の動作ではない。

また、アプリケーションの終了時においても同様に、仮想 IoT デバイスのプロセスを終了させるとプロセスが生成した TAP インタフェースが消滅する。この処理も、実験環境の終了処理に相当し、アプリケーション自体の動作ではない。

実験に使用する仮想 IoT デバイスの増加に比例して実験環境の初期化処理や終了処理に係る時間も増加する。第 7 章において行った実験においても、各クライアント上で動作させる仮想 IoT デバイスの数が 128 を超えた段階で、すべての仮想 IoT デバイスの初期化処理が終わる前に最初のシーケンス番号のメッセージが送信された。

そのため、検証においてはアプリケーションの動作と環境に関連する処理は分離出来ることが望ましい。つまり、より細かな段階に分けて仮想 IoT デバイスの制御が可能であるべきである。

8.3.3 ハードウェアエミュレータの実装に関して

本研究で提案したフレームワークでは仮想 IoT デバイスとしてハードウェアエミュレータの利用も想定している。しかし、実時間性の確保の方法や通信インタフェースの実装方法に関して考慮すべき点が存在する。

独立したハードウェアエミュレータで実時間の動作を保証することができれば、分散環境においてハードウェアエミュレータを利用して実験を行うことがより平易になる。

仮想 IoT デバイスの通信インタフェースは、ホスト OS 上でキャラクタデバイスとして認識出来るものが望ましい。このような通信インタフェースを用意することで、特に、QOMET や Meteor のようにネットワークインタフェース単位で無線通信の模倣を行うリンクエミュレータと組み合わせて利用することが容易になる。

8.3.4 無線通信の模倣に関して

本研究では、仮想 IoT デバイスを用いて並列分散環境における実験規模の拡大を実証した。さらに、仮想 IoT デバイスの通信に対して無線通信の模倣を適用することで、RPL や MPL など LLN 環境で用いることを想定したプロトコルの検証が可能となる。また、6LoWPAN をはじめとする L2 の情報に依存するプロトコルの検証のためには L2 の識別子レベルで無線通信による品質変化を模倣する必要がある。このため、第 4 章で検討した通り、Meteor を利用した無線通信のエミュレーションを実現すべきである。

また、IEEE 802.15.4 フレームをはじめとする、無線通信によるフレームを StarBED 型テストベッドの有線ネットワーク上で直接扱うことは出来ない。そこで、GUAN のインタフェース調整レイヤの実装として、仮想 IoT デバイスの生成する無線通信のフレームを有線ネットワーク上で取り扱う機構を構築することで将来的な実現を目指したい。これが実現すれば、IEEE 802.15.4 の識別子に依存する 6LoWPAN を扱うことが可能となる。

第9章　結論

本研究ではIoTを対象とした大規模実証実験環境の構築を提案した。IoTの要素技術およびネットワークシステムの大規模実証実験要素技術の調査・概観をし、IoTを対象とした大規模実証実験についての要件を整理した。既存技術の利用を検討し、既存技術で対応が困難な課題を挙げた。既存手法で対応困難な課題である、スケーラビリティについて、複数の計算機を用いることで実証実験の規模を拡大する手法を提案した。一般的な計算機においてIoTデバイスを模倣する抽象度を整理し、仮想IoTデバイスを定義した。本研究では、StarBED型テストベッドを用いてIoTを対象とした大規模実証実験環境を構築するためのフレームワークGUANを提案した。GUANに基いて実験環境を構築し、仮想IoTデバイスと複数の計算機を利用して実験規模を拡大する方法を実験した。本研究の成果によって、IoTの実証実験において複数の計算機による並列分散環境を用いる手法が有効であると結論づけた。

謝辞

本研究にあたって、主指導教員の本学 篠田 陽一教授に数多くの助言と指導を頂きました。深く感謝し心より御礼申し上げます。

研究を進めるにあたって、知念 賢一特任准教授、井上 朋哉特任助教には多くの助言と指導を頂きました。深く感謝し心より御礼申し上げます。

情報通信研究機構 三輪 信介博士、宮地 利幸博士、高野 祐輝博士、安田 真悟博士、Razvan BEURAN 博士、太田 悟氏、三浦 良介氏には研究に関する多くの助言、示唆を頂きました。心より御礼申し上げます。

WIDE Project DeepSpaceOne WG および Nerdbox Freaks の皆様には研究に関する多くの助言、示唆を頂きました。心より御礼申し上げます。

本学 宇多 仁助教には研究および生活面でお世話になりました。心より御礼申し上げます。

情報通信機構北陸 StarBED 技術センターの利用の際には中井 浩氏、廣澤 裕子氏にお世話になりました。心より御礼申し上げます。

本論文の執筆にあたり本研究室修了生 松井 大輔氏には示唆に富んだ助言を頂きました。心より御礼申し上げます。

本研究室 Muhammad Imran Tariq 氏、明石 邦夫氏、鍛治 祐希氏、村上 正太郎氏、田部 英樹氏、大野 夏希氏、向井 康貴氏、成田 佳介氏、加藤 邦章氏、岩本 裕真氏、園田 真人氏、八木 辰弥氏、可児 友邦氏には研究以外にも普段の生活においてお世話になりました。心より御礼申し上げます。

最後に、研究以外の多くの面で支えてくれた家族に心より感謝致します。

参考文献

[1] http://www.isi.edu/nsnam/ns/.

[2] https://www.planet-lab.org/.

[3] http://www.geni.net/.

[4] http://www.nsf.gov/div/index.jsp?org=CNS.

[5] http://starbed.nict.go.jp.

[6] https://www.iot-lab.info/.

[7] K. Ashton. That 'Internet of Things' Thing. http://www.rfidjournal.com/articles/view?4986, June 2009. 最初に Internet of Things という語が用いられたのは、RFID Journal の Kevin Ashton の記事によると、1999 年に行われた彼のプレゼンテーションである。

[8] R. Beuran, L. T. Nguyen, K. T. Latt, J. Nakata, and Y. Shinoda. QOMET: A Versatile WLAN Emulator. *IEEE International Conference on Advanced Information Networking and Applications (AINA-07)*, pp. 21 – 23, 2007.

[9] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard), Sept. 2011.

[10] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), Aug. 2007.

[11] T. Miyachi, K. Chinen, and Y. Shinoda. Starbed and springos: large-scale general purpose network testbed and supporting software. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, valuetools '06, pp. 1–7, New York, NY, USA, 2006. ACM.

[12] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), Sept. 2007.

[13] J.-P. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP*. Morgan Kaufmann, 2010.

[14] 岩橋, 井上, 篠田. Internet of things を対象とした大規模実証実験環境構築に関する研究. pp. 1258–1263. マルチメディア，分散，協調とモバイル (DICOMO2014) シンポジウム, July 2014.

[15] 中田. ユビキタスネットワークシミュレーション環境の構築に関する研究. PhD thesis, 北陸先端科学技術大学院大学, Mar. 2009.

[16] 知念賢一, 宮地利幸, 篠田陽一. Kuroyuri：ネットワーク実験記述言語処理系. コンピュータソフトウェア, 27(4):43–57, oct 2010.

[17] 明石, 井上, R. Beuran, 篠田. Meteor: 大規模ネットワーク実験環境における無線ネットワークエミュレータの設計と実装. 電子情報通信学会論文誌, J98-B(4), Apr. 2015. 載録予定.

本研究に関する対外発表

- 岩橋 紘司, 井上 朋哉, 篠田 陽一, *“Internet of Things* **を対象とした大規模実証実験環境構築に関する研究**” マルチメディア，分散，協調とモバイル *(DICOMO2014)* シンポジウム, pp. 1258-1263, 2014 年, 7 月. [14]