博 士 論 文

題目 モノのインターネット実現に向けた
無線通信プロトコルの研究

指導教員　門　勇一　教授

京都工芸繊維大学大学院　工芸科学研究科　電子システム工学専攻

学生番号　15822001

氏名　川本　康貴

平成 30 年 6 月 25 日提出

目次

略語一覧

AI; Artificial Intelligence
CSL; Coordinated Sampled Listening
CSMA/CA; Carrier Sense Multiple Access/Collision Avoidance
CXO; Crystal Oscillator
FA; Factory Automation
FTSP; Flooding Time Synchronization Protocol
ICT; Information and Communication Technology
IoT; Internet of Things
LSI; Large Scale Integration
NW; NetWork
MAC; Medium Access Control
PA; Plant Automation
PER; Packet Error Rate
RIT; Receiver Initiated Transmission
TDMA; Time Division Multiple Access
TSCH; Time Slotted Channel Hopping
UCoMS; Utility Infrastructure Core Monitoring System
RF; Radio Frequency
WCN; Wireless Control Network
WSN; Wireless Sensor Network

第 1 章　序論

1.1　研究背景：社会課題解決に向けたデータ主導社会の実現

現在、我が国では少子高齢化による労働人口の低下が懸念されている。例えば、人口問題研究所の資料[1]によれば、2015 年には 7000 万人程度(全体の 60%)だった日本の労働者人口(15 歳～65 歳)が、2050 年には 5000 万人(全体の 50%程度)にまで低下する。労働人口が減ると国単位でみた生産性が落ち、国力が低下する。そこで、少ない労働人口でも高い生産性や付加価値を生み出すために、ICT の活用が期待されている[2]。

今後、少ない労働力でこれまでと同等以上の生産物や付加価値の創出、作業の迅速化や精度向上のために、ICT の活用が一段と進むことが予想される。こういった兆候は日本独特のものではなく、グローバルな傾向である。例えば 2016 年の世界経済フォーラム(ダボス会議)では、現在を「ICT によって収集した多量のセンサデータや機器間のコネクティビティによる極端な自動化」を背景とした「第四次産業革命」が起こっているとしている[3]。

労働生産性を向上させるための新しい ICT として、Internet of Things(IoT)および IoT を実現するための IoT 技術が期待されている。総務省は IoT について「モノ、ヒト、サービス、情報などがネットワークを通じて大規模に連動することで新たな価値が生まれる。この内、主としてモノに着目した部分について IoT と呼ばれる」[2]と定義している。今後の IoT の発展を考えた場合、IoT を構成する IoT 機器は人間を取り巻く様々な場所に設置され、数も非常に多くなることが想定されている。例えば、アメリカの起業家である Janusz Bryzek 氏は毎年 1 兆個のセンサが活用される Trilion Sensors Universe が 2023 年までに実現するとしている[2]。IoT 機器が備える基本要素として、周辺情報を読み取るセンサ、機器を動作させるアクチュエータ、他の機器と連携するためのネットワーク、情報処理のためのコンピュータ機能がある[45]。これら基本要素の中の、IoT 機器が利用するネットワークに関しては、無線通信が有望視されている。なぜなら前述の通り、IoT 機器の設置数は非常に多くなるので、有線通信を利用したネットワークの構築は非現実的であるからである。

IoT システムのネットワークは、IoT 機器と、IoT 機器を管理する基地局からなる。IoT システムの通信は IoT 機器と基地局間でのみ実施され、IoT 機器間の通信も基地局を介して実施する。これは、ネットワーク上のすべての IoT 機器との通信経路情報を、IoT 機器が保持することが現実的ではないからである。

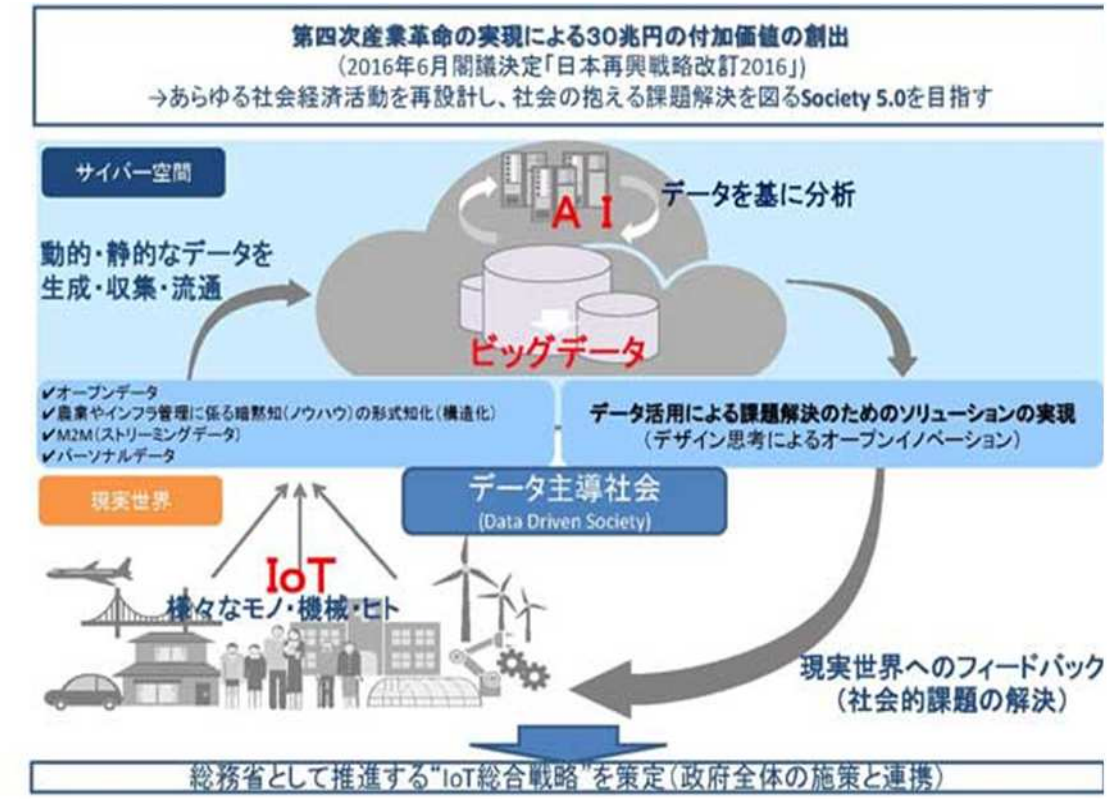

図 1 データ主導社会(経済産業省「IoT/ビッグデータ時代に向けた新たな情報通信政策のあり方」)

このような特徴をもつ IoT システムのネットワーク形態には、スター型ネットワークとツリー型ネットワークの 2 種類がある。スター型ネットワークとは、ネットワークに参加しているすべての IoT 機器が基地局と直接通信可能なネットワーク形態である。一方、ツリー型ネットワークは、基地局と直接通信できない IoT 機器もネットワークに参加できるネットワーク形態である。基地局と直接通信できない IoT 機器は、基地局と直接または間接的に通信できる他の IoT 機器にデータの中継を依頼することにより、基地局と通信する。このような中継を利用しての通信をマルチホップ通信とよぶ。

ツリー型ネットワークは、マルチホップ通信をすることで IoT 機器間の通信距離を短くできるので通信部分の回路を簡素化でき、IoT 機器を小型かつ安価に作成可能である。また、通信距離が短いと通信時の消費電力も低く、電池での長期間動作が可能となる。さらに、環境変化により通信ができなくなった場合、中継先を変更することで基地局との通信を維持できるので通信のロバスト性も高い。このようにマルチホップ通信を利用するツリー型ネットワークは IoT 機器のためのネットワーク形態に適した特徴を持つ。よって本論文では、IoT 機器のネットワーク形態は、マルチホップ通信を利用するツリー型ネットワークで

あることを想定する。

IoT 機器が使う無線ネットワーク技術の実用化は、2000 年代に実施された IEEE 802.15.4 [4]や ZigBee PRO [5]といった無線通信規格の標準化以降、しばらく停滞していた。ちょうど同時期、1998 年の電力自由化に端を発した 2000 年のカリフォルニア電力危機の発生からスマートグリッド技術に注目が集まった。スマートグリッド技術とは、電力の流れを IoT 技術によって供給・需要の両側から最適制御するためのものである。スマートグリッド技術が注目された同時期にはグリーンニューディール政策等の、スマートグリッドの実現を目的とした IoT 技術への多大な投資も行われた。その結果、2010 年代前半には、IoT 技術を利用したシステムを安価に開発する技術基盤が確立した。例えば内蔵 RAM 容量が 100 kB 以上かつコアが最も普及している ARM コア[7]で、消費電力が低い（μW オーダ）スリープ状態から、AD 変換ポート等のセンサが利用するペリフェラルからの入力で短時間(μsec オーダ)復帰が可能であるといった 1 チップマイコン[8]が登場した。他にも「ワンボード Linux」である RASPBERRY PI [9]の発売も 2010 年代初頭である。このように技術基盤が整ったことで、課題解決の手段として IoT 技術を利用したシステムを構築することが現実的になった。

このような経緯を経て、2018 年現在、IoT 技術は実用化段階に至っている。例えば、トイレの水の利用量を監視して最適化するシステム[6]や、店内の人の流れを観察して商品の配置やレジ数を最適化するシステム[10]などがある。こういったことを実現するための基礎技術は 1990 年代後半には確立していた。しかし、2010 年前後のスマートグリッドに対する設備投資が発生するまでは、リーズナブルには実現できなかった。

現在、IoT 技術で収集したセンサデータをモニタリングし、人手により状況を判断しフィードバックを実施するシステムは実用化されている。経済産業省によれば、今後は「データ主導社会」が来ると言われている[11]。データ主導社会とは、現実世界から IoT 技術で集めた様々なセンサデータをサイバー空間であるクラウド上で AI・ビッグデータ処理し、結果を現実世界にフィードバックして社会問題を解決していく社会のことをいう(図 1)。

データ主導社会の特徴はデータの収集、解析、フィードバック制御のループに人が介在しない点である。現在のシステムでは人手によるデータ解析が必要であるため、処理できるデータの量に限界があった。例えば、Utility Infrastructure Core Monitoring System (UCoMS) [14]が対象としているポンプの振動を測定することで故障を検知するようなアプリケーションの場合、いままでは半年に 1 回程度、専門の技術者がポンプ設置場所まで出向いて振動デ

ータを測定し、ポンプの健全性の判断をしていた。こういった運用の場合、センサデータを収集するための通信は既存のもので問題ない。しかし、計算機を使ってデータを解析するデータ主導社会では、人がデータを解析する場合に比べて扱うことができるデータの量が大幅に増える。よって、データ主導社会では多くのデータを収集できればできるほど、システムとして良い判断ができる可能性が高まる。例えば、前述の UCoMS アプリケーションの場合、センサの健全性判定を AI でできるようになると、ポンプの振動を 24 時間監視できるようになる。振動を 24 時間監視することにより、従来に比べて故障の兆候を早期に捉えることができ、システムをより確実に連続運用できるようになる。こういったシステムを実現するためには、バッテリ駆動で長期間連続してセンサデータを収集可能な通信プロトコルが必要である。

また、フィードバックに人が介在する現在のシステムでは、数秒以下のオーダでのフィードバック制御は不可能であった。よってシステム設計時にはフィードバック制御にかかる期間を長くとって設計をする必要があった。一般的に通信にかかる時間に比べて人手によるフィードバック内容の判断にかかる時間のほうが長い。よって、現在のシステムでは通信による遅延はシステム設計上問題にならない。しかし、データ主導社会ではフィードバック内容の判断を AI によって実施する。このため、現在のシステムでは存在する、人による判断実施のための時間が大幅に短くなる。仮に、制御情報を msec オーダでフィードバックする通信プロトコルがあれば、タイトな設計のシステム構築が可能になる。

データ主導社会が実現すると現在に比べて労働生産性が更に上がり、少子高齢化による社会課題が解決できる。我が国では 1984 年に発足した TRON プロジェクト[12]以降、IoT 技術の研究開発では世界をリードしてきた。つまり、我が国は、今後各国が直面する少子高齢化による労働人口の減少という社会的課題と、それを解決する基礎技術の両方を持つ、数少ない国と言える。このことから、我が国の研究者や技術者は、世界に先駆けてモノのインターネットを普及させてデータ主導社会を実現することで、今後グローバルに波及するであろう労働人口の減少という社会課題を他国に先行して解決し、解決方法を提示することで人類に貢献する責務がある。

1.2 課題：実用的な通信プロトコルが必要

データ主導社会を実現するためには、集めてきたデータを処理し判断する AI 技術やビッグデータ処理技術の研究開発だけでは不十分である。マルチホップ通信可能でなおかつ長期間連続してデータを集めてくることが可能な通信プロトコルや、判断結果を低遅延でフィードバックするための通信プロトコルも合わせて研究開発する必要がある。

IoT 技術を利用したシステムに限らず、システム開発は企画、要件定義、開発、移行・運用準備、運用・保守という一連のフェーズからなる[46]。以下に、システム開発の各段階の責任者の観点から、IoT システムの通信プロトコルとして望ましい特徴を述べる。

(1) 企画フェーズ/要件定義フェーズ

企画フェーズは、システムの目的とそれに付随する要件、およびシステム開発スケジュールを明らかにするフェーズである。要件定義フェーズは企画フェーズで定義されたシステムの目的を元に、システム側への要件を定義するフェーズである。どちらもシステム開発発注側の責任で実施する。

IoT 技術を使ったシステムの主要応用先の一つとしてインフラ監視システムがある。公共物であるインフラの監視システムのシステム企画者は、地方自治体や官公庁である。公共システムの開発の前に入札が行われるが、こういった公共システムの入札条件として、企画者は「通信プロトコルは標準技術を使ったシステムであること」と条件をつける場合が多い。これは、公共システムに必要な「参入障壁を作らない」という要件を満たすためである。

また、発注したシステムを今後グローバルに展開させたい場合、独自の通信技術を利用したシステムだと、国毎に違う規制等に個別対応する必要がある。これでは、スピードのあるシステム展開ができない。この場合、企画者はシステムで利用する通信プロトコルはあらかじめどの国でも利用可能であることが明らかな世界標準技術を利用するよう要求する。

以上のことから、システム発注側の観点からすると、IoT システムの通信プロトコルは標準準拠であることが望ましい。

(2) 開発フェーズ

開発フェーズは要件定義フェーズで定義された要件に対してハードウェアやソフトウェアを開発または調達し、システムを構築するフェーズである。これはシステム開発受注側の責任で実施する。

通信プロトコルの観点から見た場合、IoT システムの開発は、IoT 機器ハードウェアの開発、通信プロトコルスタックの開発、IoT アプリケーションの開発の３つからなる。一般論として、要求仕様を満たすのならば、システムの開発コストは低いほうが良く、部品調達にかかる時間は短いほうが良い。ハードウェアを標準技術で構成すると、フルスクラッチで開発したハードウェアに比べて、部品調達にかかる期間も短期間でコストも低く、開発期間も短い。例えば、OKI が発売している SmarthopSR という標準準拠のハードウェアは 2018 年現在、8000 円程度で購入できる[47]。ハードウェアを 100 台利用する IoT システムを想定した場合、標準部品である SmarthopSR を使うとハードウェアの開発コストは部品購入費用である 80 万円ですむ。この価格は 2018 年現在、新規ハードウェア開発にかかる開発コストに比べて十分低い。

さらに、通信プロトコルスタックの開発および IoT アプリケーションの開発を本格的に実施できるのは、IoT 機器ハードウェアの開発完了後である。標準技術で作られたハードウェアは市場に数多く出回っている。よって、IoT 機器ハードウェアが標準技術によって作られている場合、類似ハードウェアを使ってハードウェアの開発と並行してソフトウェアを開発できる。以上のことから、要件を満たすならば IoT 機器のハードウェアは標準技術を使って構成できたほうがシステムの開発も容易になり、コストも下がる。

以上のことから、システム開発受注側の観点からすると、IoT システムの通信プロトコルは、調達が容易である標準技術を使ったハードウェア上で実装および動作可能であることが望ましい。

(3) 移行・運用準備フェーズ/運用・保守フェーズ

移行・運用準備フェーズは、システムの設置や運用準備をするためのフェーズであり、運用・保守フェーズは設置したシステムを安定的に運用するためのフェーズである。これらはシステム運用側の責任で実施する。

システムコストの観点からすると、保守・運用フェーズにかかるコストはシステム全体にかかるコストの 60%以上になる。更に、保守運用にかかるコストのうちの 60%がシステムに対する機能拡張･機能追加に関わるコストである[48]。よって、システムコストの観点から見ると、移行・運用準備/運用・保守フェーズのコスト削減は効果が高く、重要である。

システム運用者から見た場合、IoT 機器の設置や保守・機能拡張に際して、専門知識を持たない保守運用担当者のみで対応できるシステムであることが望ましい。なぜなら、機器の追加や機能拡張の際に専門技術者が必要であればそのために別途費用を用意しておく必要があり、その分保守運用コストが高くなるからである。また、同じ理由で通信プロトコルに対する機能追加も容易である

必要がある。
以上のことから、システム運用側の観点からすると、IoT システムの通信プロトコルは専門技術者でなくても機器追加や機能拡張ができることが望ましい。

1990 年後半以降、様々な通信プロトコルが考案されている。例えば 2004 年に発表されたサーベイ論文[13]には、主なもので 25 種類の通信プロトコルが紹介されている。これらのプロトコルを利用すれば、バッテリで長期間連続動作することによって多量のデータを収集することや、無線を使った msec オーダのフィードバック制御は可能である。しかし、システムが現実的に産業界へ受け入れられるためには、IoT システムの通信プロトコルとしては技術的な要件だけではなく、上記のようなシステム開発の各フェーズの責任者の視点での要求を満足する必要がある。

以上のことから、システムが現実的に産業界へ受け入れられるためには、通信プロトコルとして以下のような特徴が必要であると整理できる。

(1) 標準準拠
システム発注者の観点からすると、今後のグローバル展開や入札の参入障壁を作らないために必要な特徴である。また、システム開発者の観点からすると、標準準拠であるとハードウェアの調達が容易になるので必要な特徴である。

(2) ハードウェアの調達容易性
システム開発者の観点からすると、システム開発コストの低減のために必要な特徴である。

(3) 低保守・運用コスト
システム運用者の観点からすると、システム保守運用、特に機能追加や機器の追加に際して専門技術者が不要であることは、保守コストを下げる観点から必要な特徴である。また、保守・運用コストはシステム全体にかかるコストの 60%を占めるので、システム全体のコストを支払うシステム発注者の観点からしても、保守・運用コストが低いことは必要な特徴である。

上記のような観点を考慮した IoT 通信プロトコルの研究開発は今まであまり行われてこなかった。よって、データ主導社会実現のために、実用的に利用できる通信プロトコルは少ないのが現状である。

1.3　本研究の目的および論文構成

本研究の目的は、1.2 節で述べた産業界に受け入れられる特徴を持った「長期間連続してデータを収集するための省電力 MAC プロトコル」や「制御に利用可能なコンパクトで使いやすいリアルタイムネットワーク(以下、NW)プロトコル」といった通信プロトコルスタックを開発し、データ主導社会を早期に実現することである。

本論文の構成は以下の通りである。

第 2 章では、実際のアプリケーション要求を元に、通信プロトコル設計に必要な要求仕様を明らかにする。

省電力 MAC プロトコルのアプリケーションはポンプの振動モニタリングシステム[14]を、リアルタイム NW プロトコルのアプリケーションは化学工場のバルブ操作システム[15]を参考にした。

第 3 章は省電力 MAC プロトコルである NES-MAC の提案である。

バッテリを使って 10 年程度の長期間連続動作が必要なインフラ監視システムでは、センサの無線通信方式として省電力 MAC プロトコルを採用する必要がある。標準の省電力 MAC プロトコルである Coordinated Sampled Listening (CSL)は、送受信間での同期がとれている場合に省電力効果がある。しかし、CSL には、通信間隔が開くと水晶クロック素子(CXO)の性能ばらつきのために同期が維持できないという課題があった。そこで本提案では CXO のばらつきを補正して長期間の同期維持を可能とする CSL 完全準拠の MAC プロトコルである NES-MAC を設計開発し、実証実験でその有効性を確認した。

第 4 章はコンパクトな制御無線向けネットワークプロトコルである NES-SORUCE の提案である。

遅延保証が必要な制御無線向け通信プロトコルでは、アクセス方式として TDMA 方式を採用している場合が多い。しかし、TDMA 方式を採用したプロトコルは実装やメンテナンス、調達が難しいという課題がある。アクセス方式として CSMA/CA 方式を採用した場合、扱いやすいが定性的には遅延保証が難しいため、制御無線としてはあまり使われていなかった。

遅延発生の原因はパケットロスであり、パケットロスの原因はパケット衝突の発生と通信環境の変化である。そこで本提案では、まず、既存の制御無線が利用される条件および要求を文献調査により明らかにした。その結果、制御無線の利用環境下で CSMA/CA 方式を使った場合のパケット衝突発生確率は、制御無線の要求から見ると影響は小さいことがわかった。よって、環境変化によるパケットロスを経路変更によって避ける方式さえ開発すれば CSMA/CA 方式

を使った制御無線 NW プロトコルが実現することがわかった。そこで我々は、バックオフ時間の短縮および 1 HOP 通信時のネットワーク ACK の省略を実施することにより、CSMA/CA 方式を利用しながら高速な経路切り替えを実現する、コンパクトな制御無線プロトコルスタックである NES-SOURCE を設計、開発し、実証実験にてその有効性を確認した。

第 5 章は第 3 章および第 4 章で提案した研究開発に対する結論および今後の課題を明らかにした。

第 2 章　通信に対する要求

2.1　省電力 MAC プロトコルに対する要求

2.1.1　背景：インフラ監視システムと無線マルチホップネットワーク

2012 年の笹子トンネルの崩壊[16]以来、日本ではインフラ監視の分野における研究開発が増加している。例えば、国土交通省は「社会インフラの監視」に焦点を当てた研究開発を推進している[17]。

インフラ監視システムは、人の立ち入りが困難な場所に設置する場合が多い。例えば橋梁の劣化を調査するためには橋の桁部分を調査する必要がある。しかし、橋梁自体が巨大である場合には人手での観察は困難で、専門の車両などを利用しての調査を実施していた。こういった場所に通信用のケーブルや電源を敷設することは難しい。よって、インフラ監視システムは設置及び保守にかかるコストが高いとみなされてきた。

IoT 通信ネットワークを構成する技術の一つに、無線マルチホップネットワーク技術がある。無線マルチホップネットワーク技術とは、IEEE 802.15.4 等の短距離無線通信方式を利用してバケツリレー方式でネットワークを構築する技術である。無線マルチホップネットワークを利用して監視システムを構築すると、通信用ケーブル敷設が不要になる。さらに、間欠動作を伴う省電力無線マルチホップネットワーク技術を利用すれば、バッテリによる長期間のモニタリングが可能となる[18]。以上のことから、インフラ監視システムの構築に省電力無線マルチホップネットワーク技術を使うと、完全ケーブルレスでシステムを構築でき、システムの設置や保守費用を抑えることができる。

2.1.2　インフラ監視向け省電力 MAC プロトコルへの要求仕様

病院や地域エネルギー供給システムといった公共性の高いインフラシステムは、安全かつ持続的に利用できることが望まれる。そのためにはシステムの中核である発電機やボイラ、ポンプなどの駆動機器、回転機器の異常を早期に検知し、機器が本格的に故障する前に修理をしてシステム全体の健全性を確保することが重要である。

機器異常の早期検知には機器の状態を常時モニタリングするシステムの導入が効果的であることがわかっている。しかしながら、このようなシステムはケ

ーブル敷設等の導入コストが高く、普及には至っていない。システム導入コスト削減には、通信配線や電源配線を無くした「システムの完全ケーブルレス化」が効果的である。完全なケーブルレス化を実現することで、配線の敷設費用が削減されるだけではなく、モニタ用センサの設置容易性も高まりシステム全体の導入コストが大幅に削減できる。

このようなポンプのケーブルレスモニタリングシステム[14]の設置環境は、パイプや他の設備といった多くの遮蔽物が存在する。このような環境で通信の接続性および信頼性を確保するためには、センシングしたデータを無線で中継するマルチホップ機能が必要である。また、電源配線をなくすためにはデータを中継する無線機も電池で動作する必要がある。

以上のことから、ポンプモニタリング用のネットワークを完全ケーブルレス化するためには、中継機も含めて電池動作可能な省電力無線マルチホップネットワークプロトコルを開発する必要があった。要求条件の調査の結果、対象のアプリケーションを実現するためのマルチホップ無線ネットワークには、第 1 章であげた「標準準拠」「ハードウェアの調達容易性」「低保守・運用コスト」に加えて「約 10 年間の連続運用」といった機能が必要であることがわかった。

インフラ監視に求められる通信頻度は概して高くない。多くとも 1 時間に 1 度程度である。例えば対象アプリケーション[14]では、多い場合でも週に 200 回程度の測定を想定している。他にも、例えば水道管漏れ検出システム[19]の監視頻度は、6 時間に 1 回程度で良いとされている。しかし、インフラ監視システムでは測定頻度が低い代わりに、システムとして「電池交換なしで 5 年～10 年の連続動作」といった事実上のメンテナンスフリーが求められる。これはインフラに設置したノードのメンテナンスには非常にコストがかかるからである。

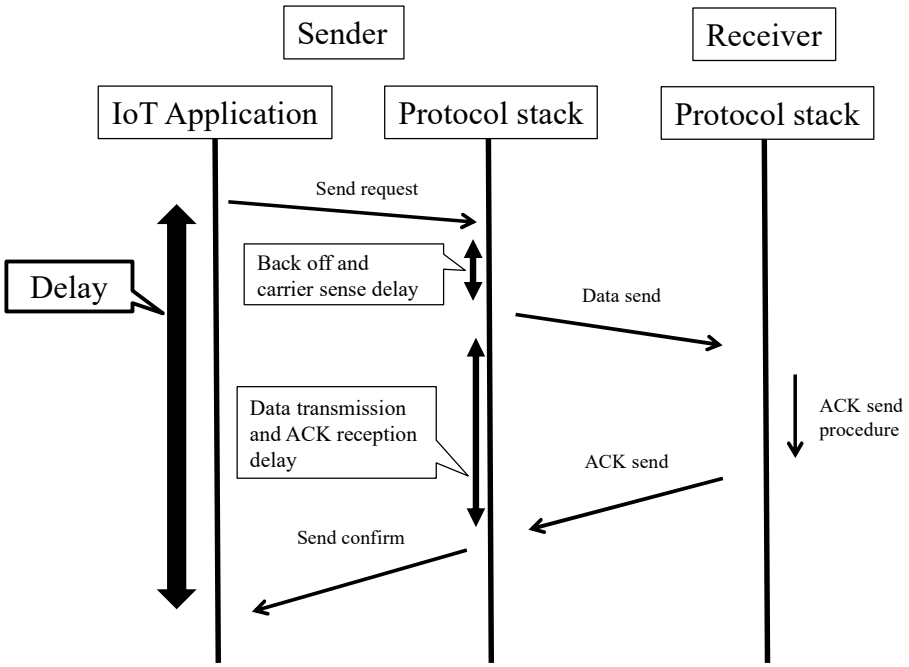

図 2 制御無線における遅延

2.2 制御無線ネットワークプロトコルに対する要求

2.2.1 背景

技術の進歩により、当初は IoT 機器間の無線通信技術として開発されてきた近距離無線ネットワーク技術(通信距離が数十 m～数百 m 程度の無線通信)は、当初の思惑に比べて広い範囲での適応が検討されている。近距離無線ネットワークの応用先として、今まで有線ネットワークで制御されてきた産業機器を、無線技術を使って制御する制御無線がある。制御無線への要求は「遅延保証」「低遅延」「低パケットエラーレート」である。

IoT 機器上で動作するソフトウェアは、IoT アプリケーションと通信プロトコルスタックの 2 つに大別できる。IoT アプリケーションは通信プロトコルスタックを利用して基地局や他の IoT 機器と通信をする。本論文での遅延とは、IoT アプリケーションが通信プロトコルスタックに対して通信要求を出してから、IoT アプリケーションが通信の成功を認識できるまで期間をいう(図 2)。遅延の要因は通信プロトコルスタックで実施するバックオフやキャリアセンス、データ送信にかかる時間、データを送信してから ACK 受信完了までの期間等がある。一般的な近距離無線ネットワーク技術として、ZigBee PRO [5]がある。

ZigBee PRO は主に屋内の家電機器を無線で相互接続するために開発されたものである。一般論として、「家電機器の設置時に、機器のネットワーク設定を

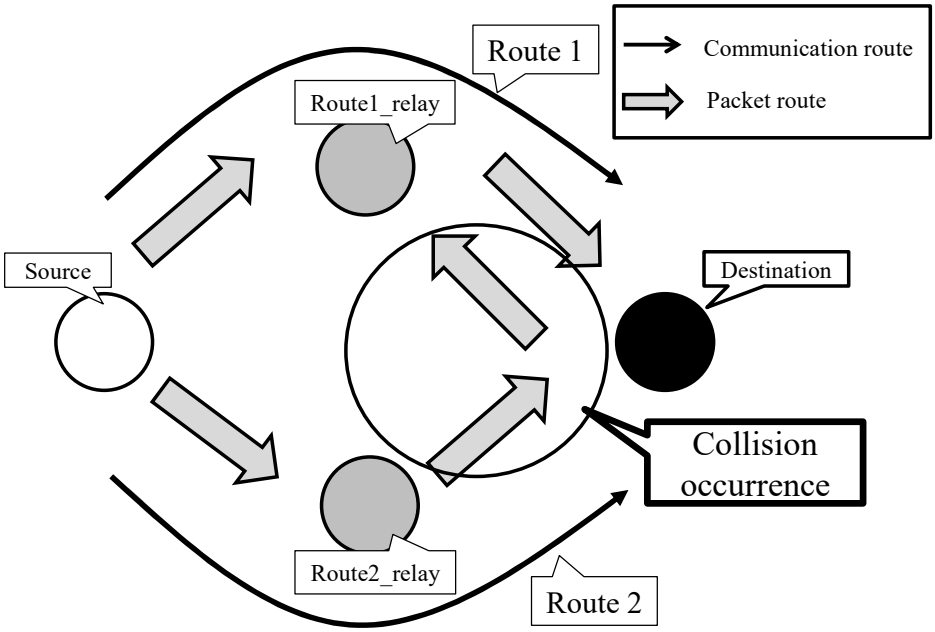

図 3 二重経路時のパケット衝突の発生

ネットワークの専門家が実施する」ということは現実的ではない。ZigBee PRO でのネットワークの設定(特にネットワークトポロジ)は機器間が協調動作して自律的に実施する。しかし、それゆえに、ZigBee PRO ではネットワークトポロジをシステム設計時に把握することができない。ネットワークトポロジが把握できない場合、データの通信経路がわからないのでデータ送信および ACK 受信完了までにかかる期間が事前に計算できず、遅延保証が困難である。制御無線システムを構築する場合、保証された遅延を考慮してシステム設計をするので、ZigBee PRO 等の既存の近距離無線ネットワーク技術で制御無線システムを構築することは難しい。

前述の通り、制御無線への要求として低遅延および低パケットエラーレートがある。データを送信している途中でパケットが消失することをパケットロスという。パケットロスが発生するとデータ再送が何度も実施されるので、その分、遅延が発生する。また、パケットロスによるデータ再送が複数回失敗すると、パケットエラーが発生する。よって、低遅延や低パケットエラーレートを実現するためにはパケットロスを減らす必要がある。無線通信は、ある周波数の電波を搬送波としてその振幅や周波数、位相を変動させることで情報を送る。受信側がこういった変動を誤って認識する確率を符号(ビット)誤り率という。パケットは符号が集まったものなので、符号誤り率が高くなるとパケットロスが発生する確率も高くなる。符号誤り率の上昇は、信号(Signal)と雑音(Noise)の比率である SN 比が受信側で許容できなくなるほど小さくなることで起こる。SN

比が小さくなる原因は、雑音が大きくなること、もしくはシグナルの出力レベルが下がることである。前者の主な原因はパケットの衝突であり、後者の原因は通信路に遮蔽物が現れるといった伝搬環境の変化である[49]。つまり、パケットロスはパケットの衝突と通信環境の変化によって発生すると言える。

制御無線プロトコルとして標準化されたものに WirelessHART [20]がある。WirelessHART では通信環境変化によるパケットロスを通信経路の二重化により防いでいる。データ通信時に 2 種類の通信経路でデータを送信することで、1 箇所の通信経路の通信環境が変化してパケットロスが発生しても、問題なく送信先へデータが届く。アクセス方式として CSMA/CA を利用している場合、同一宛先に対して複数の経路を利用して通信を開始すると、多くの場合でパケットの衝突によるパケットロスが発生する。これは複数の経路を利用したとしても最終的な到達点が同じなのでそこで輻輳が発生するからである。このような状況を示したものが図 3 である。図 3 内の黒い矢印は通信経路を、灰色の矢印はパケットが通信されていることを示す。図 3 では、通信経路 1 をつかったデータ通信の通信確認(ACK 通信)が通信路 2 のデータ通信とパケット衝突を起こしている。WirelessHART では、無線通信のアクセス制御方式として TDMA (Time Division Multiple Access)を利用してこの問題を解決している。TDMA では周辺のノード 1 つ 1 つにタイムスロットを与え、各ノードはそのタイムスロット内でのみ通信をゆるすというアクセス方式である。TDMA を利用すると原理的にパケットの衝突が発生しない。よって、経路を二重化した場合でもパケットロスは発生しない。しかし、TDMA には同じく一般的なアクセス方式である CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)に比べて以下のような課題がある。

(1) 取扱が困難

TDMA を実現するためには、通信機間で高い同期精度が必要である。

組込みソフトウェア動作環境では OS のタスクスイッチや割り込みなど、数ミリ秒の予期せぬ遅延がランダムに発生する。制御無線プロトコルは様々な構成のハードウェアで動作する必要があるので、プロトコル実装時に予め遅延を予測することは困難である。以上のことから、同期精度を 10 msec 以下で維持するプログラムの実装は容易ではない。

解決手法として例えば、組み込み向けで同期精度が高いと言われている時刻同期技術である Flooding Time Synchronization Protocol (FTSP) [50]がある。FTSP では、RF 用 LSI のドライバ内で送信フレームに時刻情報を書き込むことで前述のようなランダム遅延を排除している。通信プログラムは階層構造を持

表 1 制御アプリケーションと遅延要求の関係

Category	Class	Application	Description
Safety	0	Emergency action	Always critical e.g., Instrumented protective systems/Safeguarding systems
Control	1	Closed-loop regulatory control	Often critical e.g., Regular control loops
	2	Closed-loop supervisory control	Usually noncritical e.g., Set point manipulation for control system optimization
	3	Open-loop control	Human in loop e.g., Manual human actions on alerts
Monitoring	4	Alerting	Short-term operational consequence e.g., Short term operational consequence, event-based maintenance
	5	Logging and downloading/uploading	No immediate operational consequence e.g. history collection, sequence-of-events, preventive maintenance

Increasing importance of message timeliness

つことが一般的だが、FTSP のような階層をまたがったプログラムの作成は難易度が高い。

(2) 低遅延化が困難

TDMA 方式を採用する場合、100 msec 以下の低遅延化は難しい。なぜなら TDMA 方式ではデータ送信をあらかじめ決められたタイムスロットにしか実施できないからである。例えば MAC レイヤの通信方式として通信速度 100 kbps である IEEE 802.15.4g を利用し、最大フレームサイズ 100 byte とすると、タイムスロット 1 つの大きさは最低でも 100 byte/100 kbps = 8 msec 必要になる。通信帯域を制御無線で一般的な 25 台で分割すると、あるノードがデータ送信に使えるタイムスロットの間隔は平均で 8×25×2＝400 msec となる。実際には ACK や同期用ジッタなどが入るためにタイムスロット 1 つの大きさが 8 msec では収まらないので、送信タイムスロット間隔はさらに大きくなる。例えば WirelessHART のタイムスロット周期の初期値は 7.5 秒である。こういった課題に対処するために、通信スロット利用時のアクセス方式を TDMA 方式にするか CSMA/CA 方式にするかで区別して利用するスーパーフレームという方式がある。スーパーフレーム方式では、即応性が要求されるフレームは CSMA/CA 方式で通信をする。例えば IEEE 802.15.4 のビーコンモードはスーパーフレーム方式を採用している。しかし実装が難しく、商用ベースの実装は無い。

(3) 高い調達コスト

TDMA の同期精度を維持するためにはハードウェアの調整が必須である。例

えばリニアテクノロジの WirelessHART モジュール[21]は工場での製造時に温度変化とクロックの誤差変化を測定し、その情報をメモリに書き込むことで高い同期精度を維持している。こういった特別なハードウェアはどこでも入手できるわけではなく、システム構築のためのハードウェア調達コストは高くなる。

以上のことから、扱いやすい CSMA/CA ベースの制御無線システムが望まれている。

2.2.2 制御無線向けネットワークプロトコルへの要求仕様

Zand らの文献[22]によれば制御アプリケーションは、メッセージタイミングの重要性に応じて大きく 3 つのレベルに分類することができる(表 1)。高いレベルの要件を有するアプリケーション(例えば、生命を脅かす緊急停止装置の制御、表 1 では Safety にカテゴライズされているもの。遅延要求が 10 msec 以下)にとってはパケットロスが発生しない有線ネットワークを利用することがシステムの観点から妥当である。一方、人が介在するような、通信遅延に関する要求が低レベルであるアプリケーション(例えば、人による機器のモニタリングシステムなど。表 1 では Monitoring にカテゴライズされているもの。遅延要求が数十秒～数分程度)では、通信機能として ZigBee PRO 等の一般的な短距離無線ネットワーク通信プロトコルを利用できる。

表 2 現実的な制御無線ネットワークの技術条件

Network size	Up to 25 nodes
Packet occurrence frequency	0.1–1.0 Hz (for monitoring) 3×10^{-3} Hz (for FA or PA)
Packet error rate	0.01% order
Delay guarantees	1 s (for monitoring) 50–100 msec (for FA or PA)
Network topology	Most nodes will be located one hop from the station node and at most a distance of 3 hops will be observed.

制御無線はちょうど中レベルの要件(遅延要求が数百 msec～1 秒以下)のアプリケーションに適応できる。具体的には石油プラントのバルブ制御といったものが挙げられる。Samarasinghe らの文献[23]や Silva の文献[15]、さらに Ajith らの文献[24]、Suriyachai らの文献[25]を基に一般的に想定されている制御無線ネットワークへの要求条件を表 2 に示す。特に Silva の文献は石油プラントを対象としたバルブ操作のための現実的な制御無線システムの提案である。Ajith らの文献によれば、一般的に石油プラントといった Plant Automation (PA)が許容する遅延は 100 msec 以下、工作機械やロボットなどを利用する Factory Automation (FA)分野では 50 msec 以下である。また、Suriyachai らの文献によれば、これらのシステムが許容するパケットエラー率は 1.0×10^{-4}程度とされている。Samarasinghe らの文献や、silva の文献にも同様の条件が記されている。更に、ネットワーク規模(1 つのネットワークの構成ノードは最大 25 台程度)やホップ数(3 HOP 程度)に関しても記述されている。

CSMA/CA ベースの制御無線システムでは遅延保証が難しいため、定性的には制御無線のアクセス方式としては採用しにくい。しかし、定量的に表 2 の技術条件をクリアできれば、CSMA/CA ベースの制御無線システムを構築可能であるといえる。

2.2.3　IoT 通信に対する要求に関するまとめ

1 章では IoT システムのネットワークの特徴として「標準準拠」「ハードウェアの調達容易性」「低保守・運用コスト」が必要である旨を述べた。

本章では、インフラ監視に使われる省電力 MAC プロトコルには、技術的な省電力機能および上記の特徴に加え、10 年間の長期連続運用が可能であることを述べた。また、制御無線で利用されているアクセス制御方式である TDMA 方式は上記「ハードウェアの調達容易性」および「低保守・運用コスト」を満足することは難しいので、アクセス制御方式として CSMA/CA 方式を採用する必要があること、および制御無線の技術的条件を示した。

第 3 章　NES-MAC：モニタリング向け超省電力 MAC プロトコル

3.1　イントロダクション

2.1 節で論じたとおりインフラ監視システムを構築する場合には、IoT 通信に対して「標準準拠」「ハードウェアの調達容易性」「低保守・運用コスト」に加えて「バッテリでの 10 年程度の連続運用」といった性質が求められる。

インフラ監視システムは防犯システム等に比べて監視頻度は低くて良い場合が多い。例えば、本研究の対象である UCoMS [14]では約 1 時間に 1 回程度、水道管の水漏れ検出システム[19]では 6 時間に 1 回程度で良い。しかし、2.1 節で論じたとおり、インフラ監視システムは電源ケーブルの敷設や電池の取り替えが困難な場所に設置することが多い。よって、前述の 3 つの特性のうち、「バッテリでの 10 年程度の連続動作」はインフラ監視システムにとって特に重要な特性である。

インフラ監視に有望な、標準化された省電力無線通信方式として、Receiver-Initiated Transmission (RIT)と Coodinated Sampled Listening (CSL)がある。RIT も CSL も IEEE 802.15.4e [26]で標準化された MAC 層の技術であり、受信側が間欠動作することによって省電力を実現している。RIT と CSL は通信時の動作が違う。

RIT は非同期型の通信方式である。RIT では、受信側が受信タイミングで定期的にビーコンフレームを送信することで送信側へ自身の受信タイミングを通知する。RIT の送信側は受信側のビーコンフレームによって通知された受信タイミングに合わせてデータフレームを送信する。RIT 通信をする通信機器は相互に同期を取る必要がない。

一方、CSL は非同期型および同期型の両方に対応した通信方式である。CSL では送信側がデータフレームの送信前に Wakeup フレームを連続送信することで、受信側へデータフレームの送信タイミングを通知する。CSL を同期型で動作させる場合、Wakeup フレームの連続送信期間が非同期型で動作させる場合に比べて短くなる(CSL の詳細な動作については 3.2.1 項で詳しく述べる)。

従来では、通信の同期に使うクロックの誤差や変動により、同期通信の実施

が現実的ではないと言われていた[27]。よって、同期型の省電力通信に比べて消費電力のオーバヘッドが高い、RIT や CSL の非同期通信が利用されていた。しかし、RIT や CSL の非同期通信を使用する場合、インフラ監視システムが持つべき重要な特性である「バッテリでの 10 年程度の連続動作」の実現は難しい。ここで、もし通信時の消費電力のオーバヘッドが少ない CSL の同期通信を利用できれば現実的なサイズのバッテリで 10 年以上の連続動作が可能となる。

通常、CSL 同期通信実施のための時刻同期には一般的な水晶クロック素子(CXO)を使う。一般的な CXO は 20～30 ppm 程度の個体差や温度変動によるばらつきがある。そのため、CSL 同期通信を維持するためには 10 分～20 分に 1 度は同期のために通信をしなければならない。クロック同期処理はデータ通信実施時に実行できる。しかし、我々が対象としているアプリケーションは約 1 時間に一回の通信しか必要としていない。よって、現状のままではアプリケーションの要求に関係なく、クロック同期のためだけに通信をする必要があるということになる。これは省電力性能とシステム構築といったエンジニアリングの観点からみて問題である。

特にシステム構築の観点からすると、ノードがアプリケーションの要求仕様に関係なく省電力通信の同期維持のためだけに通信をする場合、システム設計が複雑になりシステム構築コストが上がる。この問題に対処するためには、クロック誤差を解決する必要がある。

そこで本論文では、インフラ監視に求められる特性を持つ省電力 MAC スタックとして NES-MAC を提案し、設計・実装した。NES-MAC のコアアイデアは、CSL を使った通信を用いて得られたデータをもとに、クロック誤差を修正し、通信機器間の同期を維持することである。

NES-MAC は、以下のような特徴を持つ
(1)標準準拠
NES-MAC は IEEE 802.15.4g [29]および IEEE 802.15.4e (CSL)に完全準拠している。IEEE 802.15.4g は、サブギガ帯域(920 MHz 帯域)を使った CSMA/CA ベースの MAC プロトコルとして広く使用されている。NES-MAC は、インフラストラクチャ監視システムを対象としている。ただし、NES-MAC は標準完全準拠のスタックなので、NES-MAC は、通信データの内容に関係なく期待される性能を示すことができる。ここで言う「通信データの内容」は、インフラ監視の際にセンサから得られたデータ以外のものも含む。

(2) ハードウェアの調達が容易

前述の通り、NES-MAC は IEEE 802.15.4g に完全準拠しているので世の中一般に出回っている IEEE 802.15.4g 準拠の RF 用 LSI を利用でき、低コストでシステム開発ができる。また、製造時にクロックの精度を上げるための特別な施策等は不要なので、IEEE 802.15.4g が動作する評価用ハードウェアを利用してのソフトウェア開発が可能である。

(3) 低保守・運用コスト

NES-MAC を採用したノードは、送信先ノードが NES-MAC を利用していない場合であっても CSL に準拠さえしていれば高い省電力性能を発揮できる。NES-MAC と市販の標準品とを混在してシステム構築できるので、ノードの調達や追加が容易である。結果、システムの保守・運用コストを低く抑えることができる。

(4) 10 年程度の連続動作

NES-MAC は CXO のばらつきをソフトウェアで補正する機能を持つ。よって、インフラ監視のような通信頻度が低いアプリケーションであっても同期が維持でき、通信時にオーバヘッドの少ない CSL 同期通信を維持できる。CXO のクロック誤差を修正するので、NES-MAC は IEEE 802.15.4g および CSL に完全に準拠した上で、5 年から 10 年間連続動作が可能である。

本論文では、2600 mAh の電池容量(CR123A サイズの電池 2 本程度)を使って 10 年以上の連続動作が可能であることを実機評価によって示す。

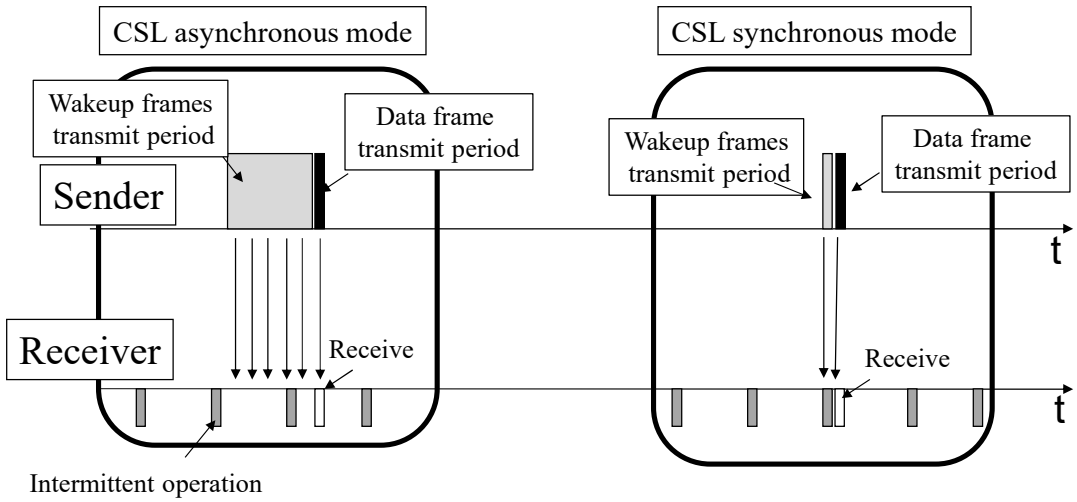

図 4 CSL の動作：受信側の受信タイミングを知ることで Wakeup フレームの送信期間を短縮

3.2 関連研究および問題点

3.2.1 Coordinated Sampled Listening システム

CSL は一般的に Low Power Listening (LPL)と呼ばれる、バッテリで長期間の連続動作を実現するための省電力技術に分類される。LPL は受信側が間欠受信動作し、送信側が受信側の間欠受信タイミングに合わせてデータを送信することで省電力を実現する通信方式である。

図 4 は CSL の動作を示す図である。CSL の送信側は、CSL 非同期通信と CSL 同期通信の 2 つの通信モードを持つ。送信側が受信側の間欠受信タイミングを知らない場合、通信は CSL 非同期通信となる。一方、送信側が受信側の間欠受信タイミングを知っていた場合、通信は CSL 同期通信になる。どちらの場合でも、送信側はデータフレームの送信前に一定期間、Wakeup フレームを連続送信する。Wakeup フレームには、データフレームの送信タイミングに関する情報 (具体的には「後何 msec 後にデータフレームを送信するか」) が入っている。受信側は間欠受信時に Wakeup フレームを受信した場合、Wakeup フレーム内の情報からデータフレーム送信タイミングを計算しデータフレーム受信待ちをする。受信側がデータフレームの受信に成功すると、受信側は送信側に対して自身の間欠受信タイミング情報を通知する。ここでいう「自身の間欠受信タイミング情報」とは、「自身の間欠受信周期」および「間欠受信時に受信した Wakeup フレーム内のデータフレーム送信タイミング情報」である。この通知はデータ

フレームの ACK フレームを使って実施される。

CSL 非同期通信時の Wakeup フレーム連続送信期間は通常、間欠受信間隔（1～3 秒程度）以上に設定する。これはデータフレーム送信にかかる期間（10 msec 程度以下）に比べて十分大きい。よってバッテリ寿命を伸ばすためには、CSL 非同期通信の頻度をできるだけ減らす必要がある。

送信側が受信側の間欠受信タイミング情報を知っている場合、通信は CSL 同期通信となる。CSL 同期通信も CSL 非同期通信時と同様に、データフレーム送信前に Wakeup フレームを連続送信する。ただし、CSL 同期通信時の Wakeup フレーム送信期間は CSL 非同期通信時に比べて十分短い。たとえば NES-MAC の実装では CSL 同期通信時の Wakeup フレーム送信期間は約 20 msec である。CSL 同期通信完了時にも、受信側は送信側へ自身の間欠受信タイミングを通知する。CSL 同期通信時にデータフレーム送信に失敗した場合、送信側は受信側の間欠受信タイミング情報を消去し、次回の通信は CSL 非同期通信でする。

3.2.2 水晶クロック素子の誤差に着目した LPL 方式と課題

送受信の時刻同期用クロックの精度が高く、誤差がない場合は一度 CSL 非同期通信で間欠受信タイミング情報を取得すれば CSL 同期通信が失敗することはない。しかし、センサネットワークで利用するノードの時刻同期用クロックは、精度が比較的低い(20～30 ppm 程度)CXO を利用することが普通である。このため、通信間隔が数十分以上になると CSL 同期通信に失敗してしまい、通信時にオーバヘッドの大きい CSL 非同期通信をすることになる。

クロック誤差に着目した LPL プロトコルの研究は、大きく分けて CXO の仕様に基づく「想定誤差補正」方式と「実測誤差補正」方式の 2 つの方式に分けることができる。

「想定誤差補正」方式では WiseMAC [30]が広く知られている。WiseMAC はデータフレーム送信前の Wakeup 信号送信期間(WiseMAC ではプリアンブル送信期間)をデータ送信間隔とクロック誤差を参照して求める。WiseMAC はデータ送信間隔が長くなると Wakeup 信号送信期間が長くなってしまい、フレーム送信時のオーバヘッドが大きくなるという課題がある。WiseMAC の文献によれば、フレーム送信時のオーバヘッドは式(1)で表すことができる。式(1)内のθ は CXO のクロック誤差を示し、$Interval_{dt}$は通信間隔を示す。$Interval_{wakeup}$は受信側の間欠受信間隔を示す。

$$Overhead_{wisemac} = min(4\theta \times Interval_{dt}, Interval_{wakeup}) \qquad (1)$$

例えば間欠受信間隔が 3 秒でクロックの誤差が 30 ppm の CXO を利用している場合、WiseMAC ではフレーム送信間隔が 7 時間を超えると Wakeup フレーム送信期間が間欠受信期間を超えてしまい、フレーム送信にかかるオーバヘッドが、時刻同期の確立していない場合と同じになる。

「実測誤差補正」方式とは、システム稼働時に通信相手との誤差を測定してクロック補正をする方式である。この方式には RATS [31]、PW-MAC [32]、および OpenWSN [33]で採用されている David らの方式 [34]がある。RATS および PW-MAC はどちらもビーコンを送信して誤差を補正する方式であるため、10 年程度の長期間動作の観点からは問題がある。David らの方式は本論分の提案方式と同じく、ビーコンを利用することなく誤差を補正する方式である。しかし、David らの方式は TDMA のタイムスロット開始時刻から送信側のデータ送信開始時刻までの期間([33]内では TsTxOffset と規定)のずれを利用してクロックの誤差補正をしている。すなわち、相対時刻を利用したクロック補正をしている。このため、David らの方式は、通信するすべてのノードの間欠受信周期が同じでなければ利用できない。しかし、本提案はノード内の絶対時刻を利用してクロックの誤差補正をする。よって、間欠受信周期の違うノード間でも同期が維持できる。実際にインフラ監視のためのセンサネットワークを構築した場合、ノードに接続するセンサの消費電力や、接続できるバッテリサイズ応じてノードの間欠受信周期を調整する必要がある。したがって、間欠受信周期が同じであることを前提とした David らの方式は省電力型のセンサネットワークに不適である。

表 3 クロック素子の性能比較

	CXO: (RIVER ELETEC CORPORATION: TFX-03)	TCXO: (NDK:NT2016SD)	OCXO: (TAMADEVICE: SCOCXOLWT)
Size(mm)	2.0 × 1.2 × 0.6	2.0 × 1.6 × 0.8	18.5 × 13.2 × 1.0
Operation current	0.2 μA	1.5 mA	80 mA
Power-supply voltage	1.7–3.3 V	1.7–3.3 V	3.3 V
Clock accuracy	30 ppm	1 ppm	0.075 ppm

3.2.3 クロックの高精度化と課題

ハードウェアクロックを精度の低い CXO から精度の高い TCXO (温度補正回路付 CXO：精度数 ppm)や OCXO (恒温槽入り CXO：0.1 ppm 以下)へ変更すると同期精度も上がる。表 3 に、一般的なハードウェアクロックのサイズと性能を示す。表 3 を見ると、TCXO または OCXO の動作電流(mA オーダー)が CXO の動作電流(μA オーダ)よりもはるかに大きいことがわかる。

インフラ監視システムでは、バッテリを使用して 10 年間継続して動作させる必要がある。この場合、ノードの平均動作電流は μA のオーダである必要があるので、TCXO と OCXO はインフラ監視システムには使用できない。WirelessHART は、ノード製造時にクロック補正情報を 1 つずつ測定してノードに書き込むことにより、低精度の CXO を使用して高い同期精度を実現している。しかし、この方法では、ノード製造時に補正情報を取得する必要があり、コストが高くなるという課題がある。また、システム設計の観点からは、他の方法のノードとの互換性がない(他社製品ノードと通信時には省電力機能を発揮できない)という課題がある。

3.2.4 標準の省電力無線通信方式

CSL 以外で標準化された省電力無線通信方式として、IEEE 802.15.4e で規定されている RIT がある。RIT は受信側がビーコンを利用して自身の受信タイミングを周囲へ通知し、送信側はそのタイミングに合わせてフレームを送信する通信方式である。RIT の動作概要を図 5 に示す。

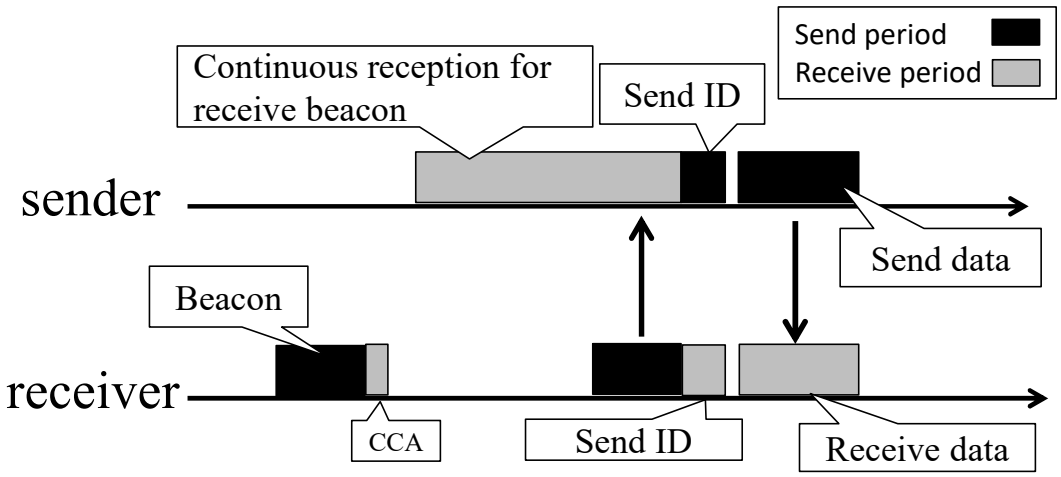

図 5 RIT の動作

表 4 RIT と CSL を比較する際のパラメタ

AT	Transmit current (A)	49 mA
AR	Receiving current (A)	28 mA
AI	Idle current (A)	1.7 μA
λ_{Link}	Request frequency for link establishment (s^{-1})	-
T	Wakeup interval (s)	3 sec
T_{Data}	Data transmit period (s)	3.2 msec
T_{RIT_ID}	ID transmit period in RIT (s)	1.28 msec
T_{RIT_CS}	Carrier sense period in RIT (s)	10 μsec
T_{CSL_wf}	Wakeup frame receive period in CSL (s)	2 msec
T_{CSL_ws}	Wakeup frames transmit period in CSL synchronous mode (s)	20 msec
T_{CSL_CS}	Carrier sense period in CSL (s)	2 msec
N	The number of peripheral nodes	25

表 5 各方法における単位時間当たりの通信動作割合

RT_{RIT}	Transmission operation ratio per unit time in RIT
RR_{RIT}	Receive operation ratio per unit time in RIT
RI_{RIT}	Idle operation ratio per unit time in RIT
RR_{CSL}	Receive operation ratio per unit time in CSL
RT_{SCSL}	Transmission operation ratio per unit time in CSL synchronous mode
RI_{SCSL}	Idle operation ratio per unit time in CSL synchronous mode
RT_{ACSL}	Transmission operation ratio per unit time in CSL asynchronous mode
RI_{ACSL}	Idle operation ratio per unit time in CSL asynchronous mode

CSL ではデータフレーム送信前に Wakeup フレームを連続送信したが、RIT ではデータフレーム送信前に送信側がビーコン受信のために受信待ちをする。送信側は送信先ノードのビーコンを受信後、データフレームを送信する。藤原らの文献[27][28]によれば、RIT は CSL 非同期通信に比べ、端末間の通信頻度が高い場合でも電波占有時間が増えない点、および RIT は非同期通信方式であるので CXO のばらつきによる影響を受けない点が CSL に比べて優れている。

インフラ監視アプリケーションは通信頻度が低いので、通常の CSL を使うと通信が CSL 非同期通信になる。藤原らの文献では CSL 非同期通信と RIT を消費電力と通信リンク確立成功率の観点から比較している。比較の結果、インフラ監視アプリケーションの通信頻度(1 時間に 1 回程度)では、消費電力および通信リンク確立成功率の観点から見ると CSL 非同期通信よりも RIT のほうが優れている。しかし、藤原らの文献では CSL 同期通信と RIT の比較はされていない。そこで、CSL と RIT のどちらがインフラ監視に最適であるかを明らかにするために、RIT と CSL 非同期通信、CSL 同期通信を消費電力および通信リンク確立成功率の観点から、理論計算により比較した。

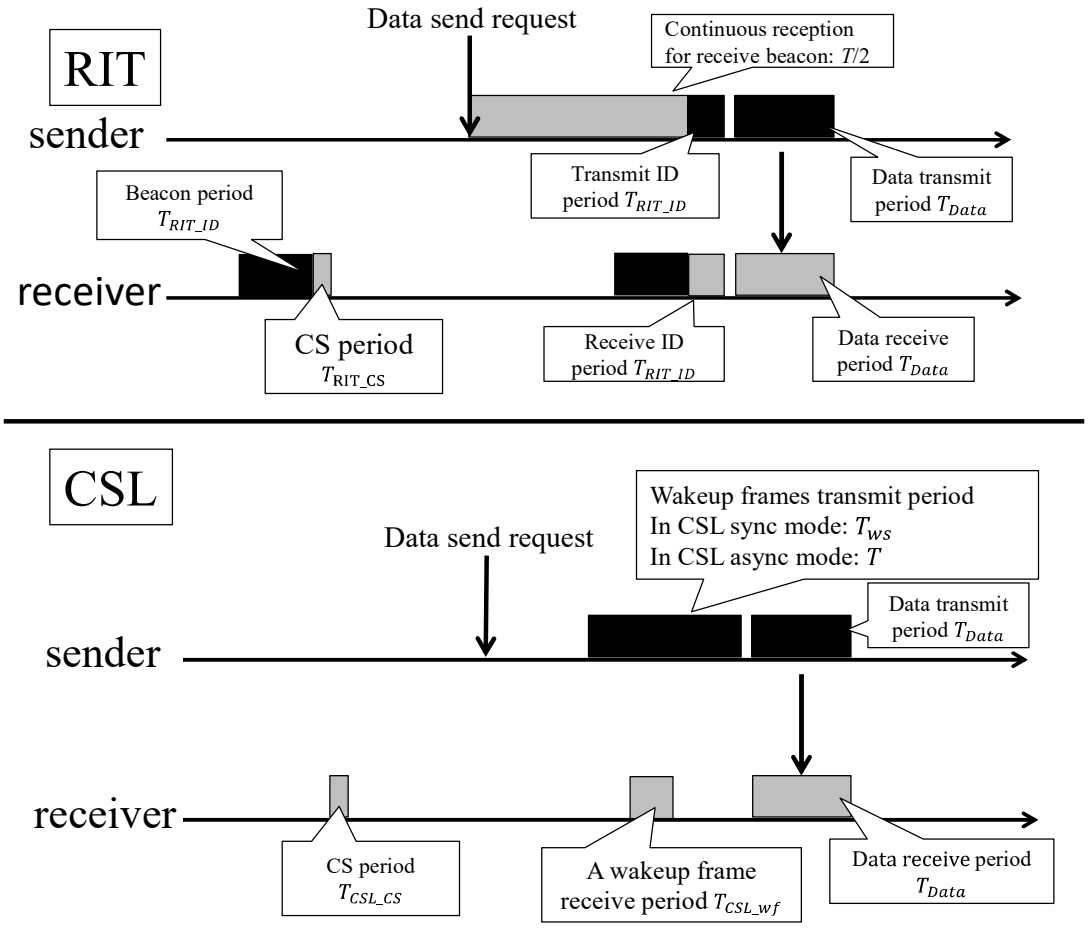

図 6 RIT および CSL の動作

比較のための計算モデルは、藤原らの文献のものを利用した。計算モデルに利用するパラメタを表 4、表 5、及び図 6 に示す。これらのパラメタにおいて RIT にに関連するパラメタは藤原らの文献のものを利用し、CSL に関するパラメタおよび消費電流値は OKI 製 MH-920-Node-232 [35]の実装時の値を利用した。

また、通信リンク成功確率の計算に用いられる周辺ノード数は藤原らの論文で書かれている 100 台ではなく、インフラ監視のシステム環境をもとにした論文[23]での値である 25 台を利用した。

3.2.4.1 理論計算による消費電力の比較

RIT、CSL 非同期通信、CSL 同期通信を利用し、表 4 で示す性能を持つノードを 1 時間動作させるのに必要な放電容量(D_{RIT}, D_{ACSL}, D_{SCSL})を式(2)～(4)に示す。単位は mAh である。

$$D_{RIT} = RT_{RIT} \times AT + RR_{RIT} \times AR + RI_{RIT} \times AI \quad (2)$$

$$D_{ACSL} = RT_{ACSL} \times AT + RR_{CSL} \times AR + RI_{ACSL} \times AI \quad (3)$$

$$D_{SCSL} = RT_{SCSL} \times AT + RR_{CSL} \times AR + RI_{SCSL} \times AI \quad (4)$$

更に、RIT、CSL 同期通信、CSL 非同期通信の 1 時間当たりの送信時間、受信時間、アイドル時間を以下の式(5)～(12)に示す。

$$RT_{RIT} = \frac{T_{RIT\_ID}}{T} + \lambda_{Link} \times (T_{Data} + T_{RIT\_ID}) \quad (5)$$

$$RR_{RIT} = \frac{T_{RIT\_CS}}{T} + \lambda_{Link} \times (T_{Data} + T_{RIT\_ID} + \frac{T}{2}) \quad (6)$$

$$RI_{RIT} = 1 - \left(RT_{RIT} + RR_{RIT} \right) \quad (7)$$

$$RR_{CSL} = \frac{T_{CSL\_CS}}{T} + \lambda_{Link} \times (T_{Data} + T_{CSL\_wf}) \quad (8)$$

$$RT_{SCSL} = \lambda_{Link} \times (T_{Data} + T_{CSL\_ws}) \quad (9)$$

$$RI_{SCSL} = 1 - \left(RT_{SCSL} + RR_{CSL} \right) \quad (10)$$

$$RT_{ACSL} = \lambda_{Link} \times (T_{Data} + T) \quad (11)$$

$$RI_{ACSL} = 1 - \left(RT_{ACSL} + RR_{CSL} \right) \quad (12)$$

各方式を利用した際の 1 時間当たりの放電容量D_{RIT}，D_{ACSL}，D_{SCSL}から、各方式を利用した機器を 10 年間維持動作させるのに必要な電池容量D_{10Y_RIT}，D_{10Y_ACSL}，D_{10Y_SCSL}は以下の式で計算できる。

$$D_{10Y\_RIT} = D_{RIT} \times 24 \times 365 \times 10 \quad (13)$$

$$D_{10Y\_ACSL} = D_{ACSL} \times 24 \times 365 \times 10 \quad (14)$$

$$\mathrm{D}_{10Y\_SCSL} = \mathrm{D}_{SCSL} \times 24 \times 365 \times 10 \quad (15)$$

式(2)～式(15)より RIT、CSL 非同期通信、CSL 同期通信を 10 年間動作させる

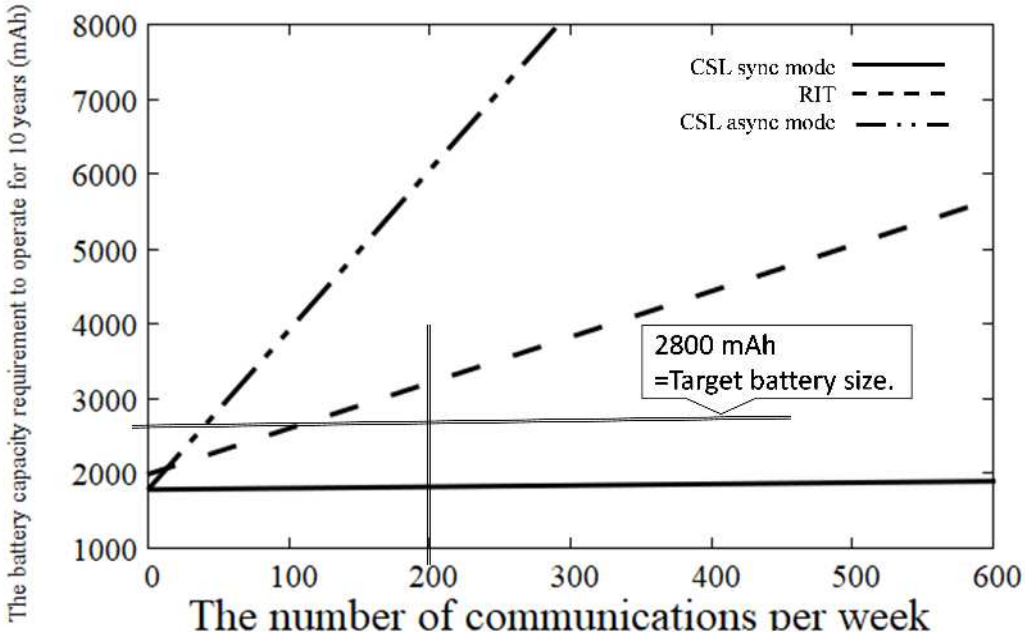

図 7 1 週間あたりの通信回数と 10 年間連続動作に必要なのバッテリ容量の関係：CSL 同期モードのみが想定しているバッテリ容量で 10 年間動作可能

のに必要な電池容量を比較したグラフが図 7 である。縦軸が 10 年間連続動作に必要な電池容量であり、横軸が 1 週間あたりの通信回数である。

例えば UCoMS [14]のシステムの要求である「1 時間に 1 回程度のフレーム通信」を実施したとすると、1 週間当たり約 200 回のフレーム送信をすることになる。1 週間に 200 回のフレーム送信を実施した場合に、10 年間の連続動作に必要な電池容量は、式（2）～（15）と表 2 の値を使った計算により、CSL 非同期通信を利用した場合は 6051 mAh、RIT を利用した場合は 3215 mAh、CSL 同期通信を利用した場合は 1821 mAh と推測できる。この計算結果から、 RIT および CSL 非同期通信を利用した場合は、UCoMS [14]で想定する電池容量（CR123A を 2 本。2800 mAh）では 10 年間連続動作できない。しかし、同期維持ができて CSL 同期通信ができると、想定する電池容量で 10 年間連続動作できると推測できる。

表 6 実験時の条件

Wireless communication system	IEEE 802.15.4g
PHY rate	100 kbps
CSL wakeup interval	3 sec
Wakeup frames transmit period in CSL synchronous mode	20 msec

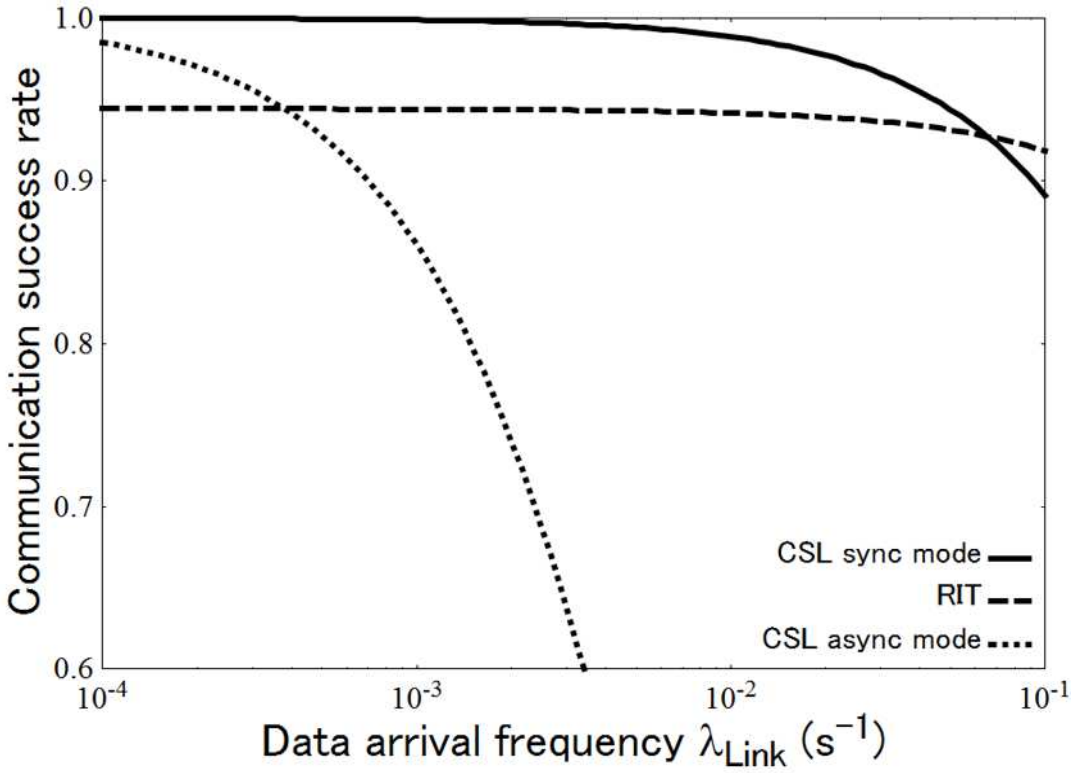

図 8 通信成功率の比較：通信頻度が 15 秒に 1 回以上になると、CSL 同期モードの通信成功率が最も高くなる

3.2.4.2　通信リンク確立成功率の理論的比較

通信リンク確立要求がポアソン分布に従って発生し、ピュアアロハ方式の理論式[36]と同様の考え方で通信が衝突しない確率、すなわち通信リンク確立成功率の理論式を藤原らの文献を元に導出した。結果を図 8 に示す。導出時の条件を表 6 に、導出式を式(16)～(18)に示す。

$$S_{RIT} = e^{-N\left\{2\cdot\lambda_{Link}\cdot\left(2\cdot T_{RIT\_ID}+T_{Data}\right)+\frac{s\cdot T_{RIT\_ID}+T_{Data}}{T}\right\}} \quad (16)$$

$$S_{CSL\_ASYNC} = e^{-2N\cdot\lambda_{Link}\cdot\left(T+T_{Data}\right)} \quad (17)$$

$$S_{CSL\_SYNC} = e^{-2N\cdot\lambda_{Link}\cdot\left(T_{CSL\_ws}+T_{Data}\right)} \quad (18)$$

RIT および CSL 非同期通信利用時の通信リンク確立成功率S_{RIT} およびS_{CSL_ASYNC}は、藤原らの文献[27]の式を利用した。CSL 非同期通信の通信リンク成功確率S_{CSL_SYNC}である式（18）は、CSL 非同期通信の通信リンク確立成功率である式（17）内のリンク確立にかかる期間$T + T_{Data}$から CSL 同期通信時のリンク確立にかかる期間である$T_{CSL_ws} + T_{Data}$に変更することで導出した。

図 8 によれば、通信リンク確立要求(λ)が 0.065 程度(約 15 秒に 1 回、1 週間当たり約 40000 回のフレーム送信)以上になると RIT のほうが CSL より通信リンク確立成功率が高くなる。インフラ監視アプリケーションの通信頻度は 1 時間に 1 回程度と低い。以上のことから、通信リンク確立成功率の観点から見ると、インフラ監視には CSL 同期通信が RIT よりも有効であるといえる。

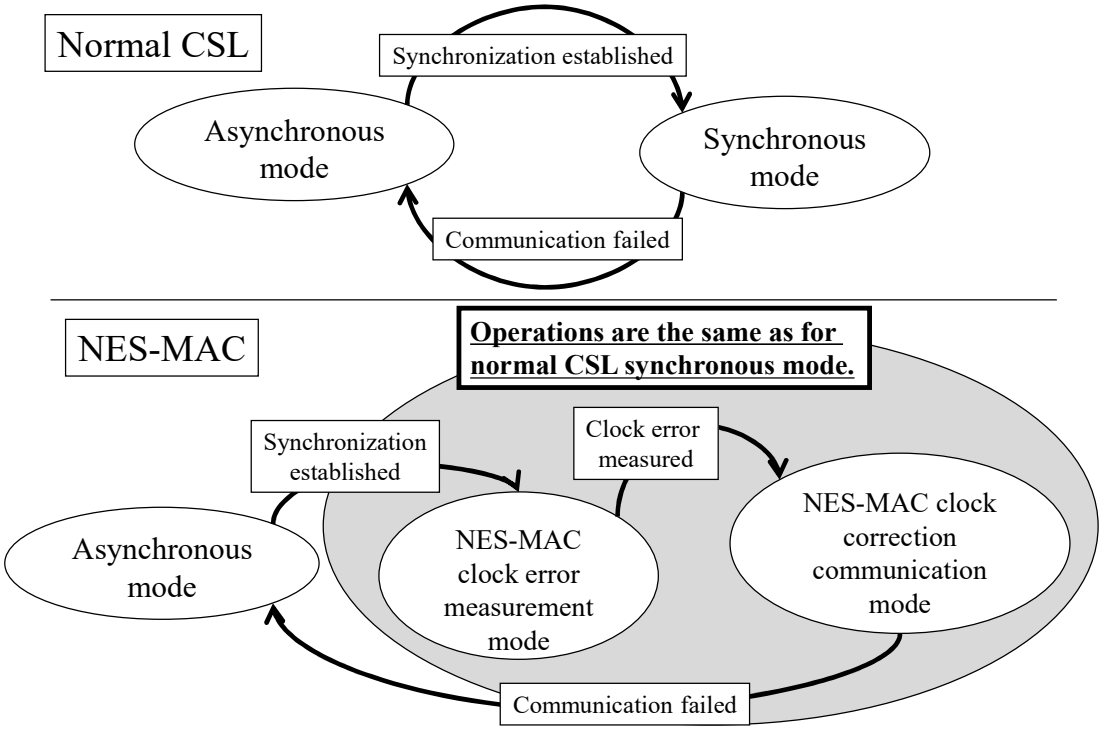

図 9 通常の CSL と NES-MAC の状態遷移図

3.3 NES-MAC の設計

3.3.1 NES-MAC の動作

CSL はフレーム送信時にはまず CSL 非同期通信を実施し、送信先との同期が確立するとそれ以降のフレーム送信は CSL 同期通信によって実施する。CSL 同期通信時に通信が失敗すると、同期が外れたと判断してフレーム送信時には CSL 非同期通信をするようになる(図 9 上)。

NES-MAC は送信先との同期が確立していない間は CSL と同じく CSL 非同期通信をする。送信先との同期が確立するとクロック補正通信をする。具体的には CSL 非同期通信が確立した際の送信先ノードからの同期情報とデータ送信間隔からクロックの誤差を算出する。クロック誤差が算出できたら通信の度にクロック補正をしながら CSL 同期通信をする、クロック補正済み同期通信をする(図 9 下)。

通信失敗時には従来の CSL と同じく非同期通信に戻る。NES-MAC のクロック補正方式は「送信元が送信先のクロックに合わせる」方式なので、通信先が NES-MAC ではない従来の CSL であっても省電力効果がある。また、NES-MAC は CSL に完全準拠しているので、送信元が従来の CSL であった場合でも問題

```
/* CslAsyncDataTransmit() and CslSyncDataTransmit() return the true wakeup timing as a return value. */
/* When CslSyncDataTransmit() is executed, the frame is transmitted at the timing of the argument. */

/* *CSL async mode start* */
async_true_wakeup_timing = CslAsyncDataTransmit() ;    ...(1)
/* *CSL async mode end* */

/* *NES-MAC clockerror measurement mode start* */
temp_wakeup_timing = async_true_wakeup_timing;
every WAKEUP_INTERVAL
   temp_wakeup_timing += WAKEUP_INTERVAL;    ...(2)
   if (GetTransmissionRequest()) break;
end loop:
sync_true_wakeup_timing = CslSyncDataTransmit (temp_wakeup_timing);
wakeup_timing_diff  = sync_true_wakuep_timing - temp_wakeup_timing;   ...(3)
clock_error = wakeup_timing_diff/(temp_wakeup_timing - async_true_wakeup_timing);   ...(4)
/* *NES-MAC clockerror measurement mode end* */


/* *NES-MAC clock correcition communication mode start* */
temp_wakeup_timing = sync_true_wakeup_timing;
every WAKEUP_INTERVAL
   temp_wakeup_timing += wakeup_interval;
   if (GetTransmissionRequest()){
      correction_val = (temp_wakeup_timing - sync_true_wakeup_timing)*clock_error;   ...(5)
      sync_true_wakeup_timing = CslSyncDataTransmit (temp_wakeup_timing+correction_val);   ...(6)
   }
```

図 10 NES-MAC の擬似コード

なく省電力通信が可能である。

3.3.2　クロック誤差算出方法

NES-MAC はクロック補正通信時にクロック誤差を算出する。クロック誤差の算出方法を、図 10 および図 11 で説明する。図 10 は NES-MAC の擬似コードであり、図 11 はクロック補正通信時の NES-MAC の通信シーケンスである。

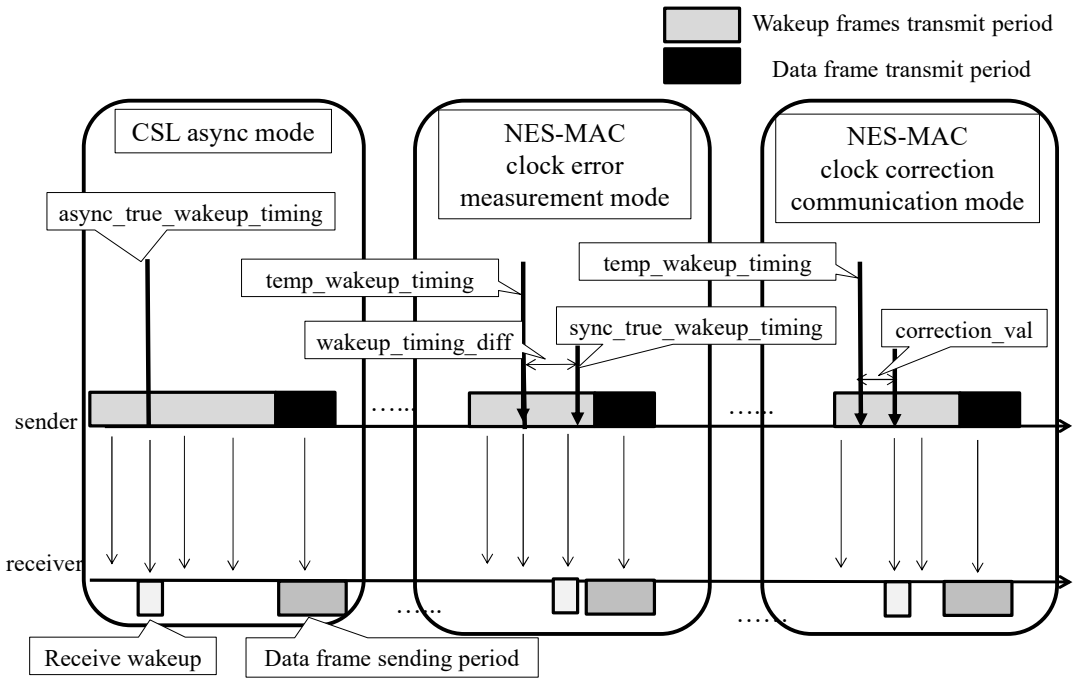

図 11 NES-MAC：クロック補正通信時の通信シーケンス

3.2.1 節で論じたとおり、LPL(CSL および NES-MAC)は、データフレーム送信成功時の ACK 受信時に、受信側の間欠受信タイミングを検出する。CSL 非同期通信(図 10(1))でデータを送信した後、NES-MAC は、NES-MAC クロックエラー測定状態に状態を変更する。NES-MAC は、図 10(1)および(2)で検出した受信側の間欠受信タイミングおよび間欠受信間隔に基づいて受信側が受信状態になる時刻を常に計算する。上位層から NES-MAC へのデータ送信の要求があった場合、NES-MAC は、計算された受信側の間欠受信時刻に CSL 同期通信を使ってデータを送信する。NES-MAC は、CSL 同期通信の結果として取得された間欠受信タイミングと、自身が推定した受信側の間欠受信タイミングの差を計算する(図 10(3))。NES-MAC は、図 10(3)で計算されたタイミング差に基づいて、送受信間のクロック誤差を計算する(図 10(4))。その後、NES-MAC は、NES-MAC クロック補正通信状態へ状態遷移する。上位レイヤからのデータ送信要求があると、NES-MAC は受信側の間欠受信時刻を計算する。その後、NES-MAC は図 10(4)で計算されたクロック誤差を使用して受信側の間欠受信時刻を修正し(図 10(5))、データフレームを送信(図 10(6))する。

通常の CSL では、データ送信間隔が長くなるにつれて、クロックエラーによる同期ずれが Wakeup フレーム通信期間よりも長くなる。その結果、CSL 同期通信通信が失敗し、通信状態が CSL 同期通信から CSL 非同期通信に戻ってし

まう。つまり、CSL 同期通信を長期間維持できない。しかし、NES-MAC では、通信の度にフレーム送信開始時刻が補正される。このため、CSL 同期通信を失敗しない。よって、データ送信間隔が長くなっても、NES-MAC の場合にはデータ通信に成功するので CSL 同期通信を維持できる。

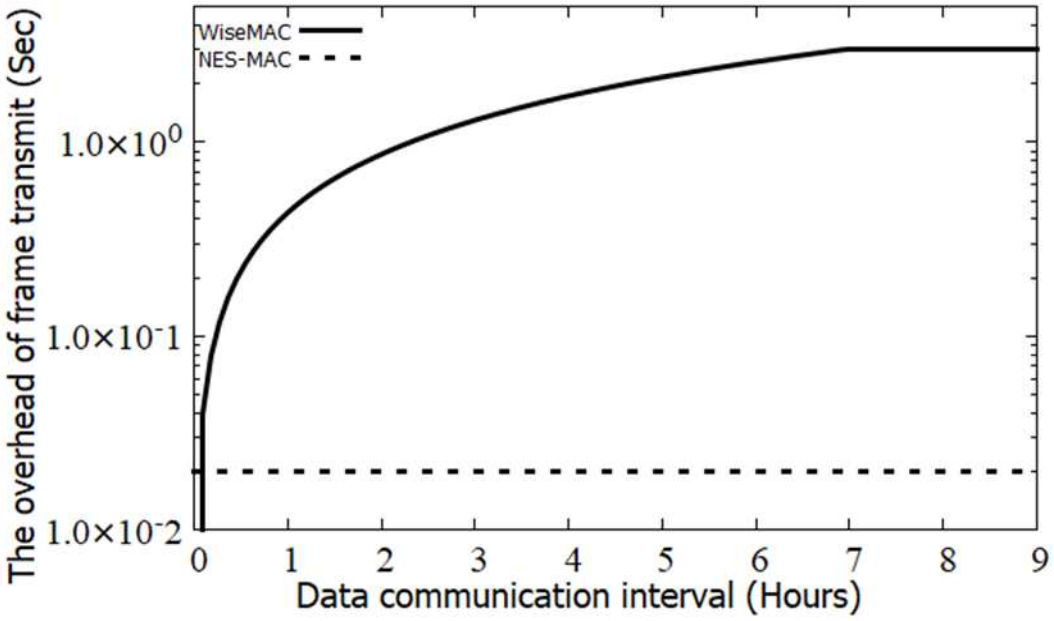

図 12 WiseMAC と NES-MAC のオーバヘッドの比較。NES-MAC のオーバヘッドは一定だが、WiseMAC のオーバヘッドは、通信間隔の増加につれて増加

3.4 評価

3.4.1 従来技術との比較

WiseMAC と NES-MAC はどちらも CSMA/CA ベースの MAC プロトコルである点、LPL ベースの省電力 MAC 技術である点、および、CXO の誤差に注目している点から似ているところが多い。よって、WiseMAC と NES-MAC をデータ通信時のオーバヘッドの観点から定量的に比較評価をした。

WiseMAC と NES-MAC をフレーム送信時のオーバヘッドの観点から比較した結果を図 12 に示す。縦軸はフレーム当たりのオーバヘッドの長さを示し、横軸はデータ通信間隔を示す。 WiseMAC の場合、データ通信時のオーバヘッドはプリアンブル送信期間である。オーバヘッドは、式(1)を使用して計算することができる。NES-MAC の場合、データ通信時のオーバヘッドは、CSL 同期通信時の Wakeup フレームの送信期間である。CSL 同期通信時の Wakeup フレーム送信期間は表 6 の 20 msec を利用した。また、CXO のクロック誤差は 30 ppm とした。

図 12 より、NES-MAC ではデータ通信間隔が大きい場合においてもデータ通信時のオーバヘッドは変化しないことがわかる。しかし、WiseMAC では、データ通信間隔が広がるとデータ通信時のオーバヘッドが大きくなる。UCoMS では、週 200 回程度の通信があると想定している。この場合、平均通信間隔は 0.84 時間になるので WiseMAC 利用時のデータ通信時のオーバヘッドは約 360 msec となる。

3.2 節の式および値を利用して通信間隔が 0.84 時間であるときに必要な電池容量を計算した。NES-MAC を使った場合、10 年間連続動作に必要な電池容量は 1821 mAh である。一方、WiseMAC を使った場合に必要な電池容量は 2330 mAh である。WiseMAC 利用時に必要な電池容量は NES-MAC 利用時よりも 30%多い。これは有意な差であると言える。

更に、WiseMAC を利用する際は受信側の CXO のクロック誤差が予めわかっていなければ機能しない。受信側のクロック誤差が想定よりも大きい場合はデータ通信が失敗し、省電力効果が達成できない。一方 NES-MAC は通信中にクロック誤差を測定する方式である。よって、受信側のクロック誤差情報がなくても同期通信を維持でき、省電力効果を発揮できる。以上のことから、インフラ監視システムに於いては、NES-MAC は WiseMAC に比べて優れたパフォーマンスを示すと言える。

3.4.2 実装による評価

3.4.2.1 実装環境

提案手法の検証のため、OKI 製 MH920-Node-232 上で実装し評価した。MH920-Node-232 は 32 ビットマイコンである Coretex-M3 を搭載した IEEE 802.15.4g 対応のセンサノードである。ソフトウェアの開発は組み込み OS として FreeRTOS [37]を、開発言語は C を採用した。その他、実験条件は表 6 に従った。ただし、オシロスコープでの同期信号等の撮影時には、簡単のため、Wakeup interval は 1 秒に変更して撮影をした。NES-MAC の動作は Wakeup interval によらないためこの変更が実際の動作に影響することはない。

また、通常の CSL のためのプログラムのコードサイズは約 900 ステップであった。NES-MAC の実装のために、CSL のためのプログラムコードへ約 30 ステップのプログラムコードを追加した。以上のことから、CSL と NES-MAC のプログラムコードサイズの観点から見た違いは約 3%である。プログラムの複雑さの観点から、通常の CSL と NES-MAC の間には大きな違いは無いと言える。

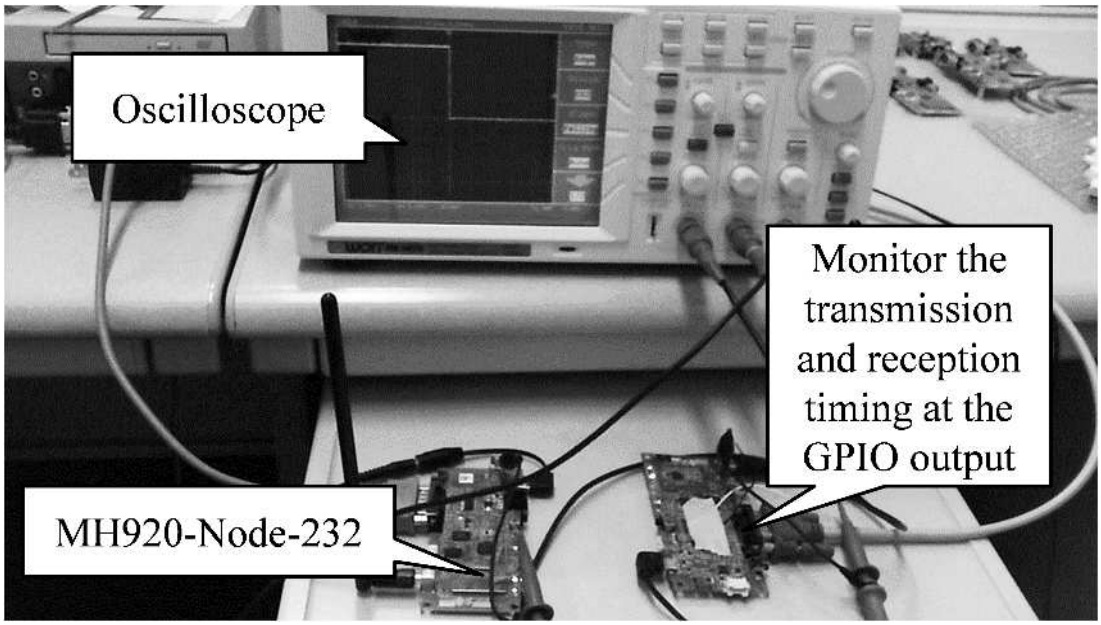

図 14 実験環境

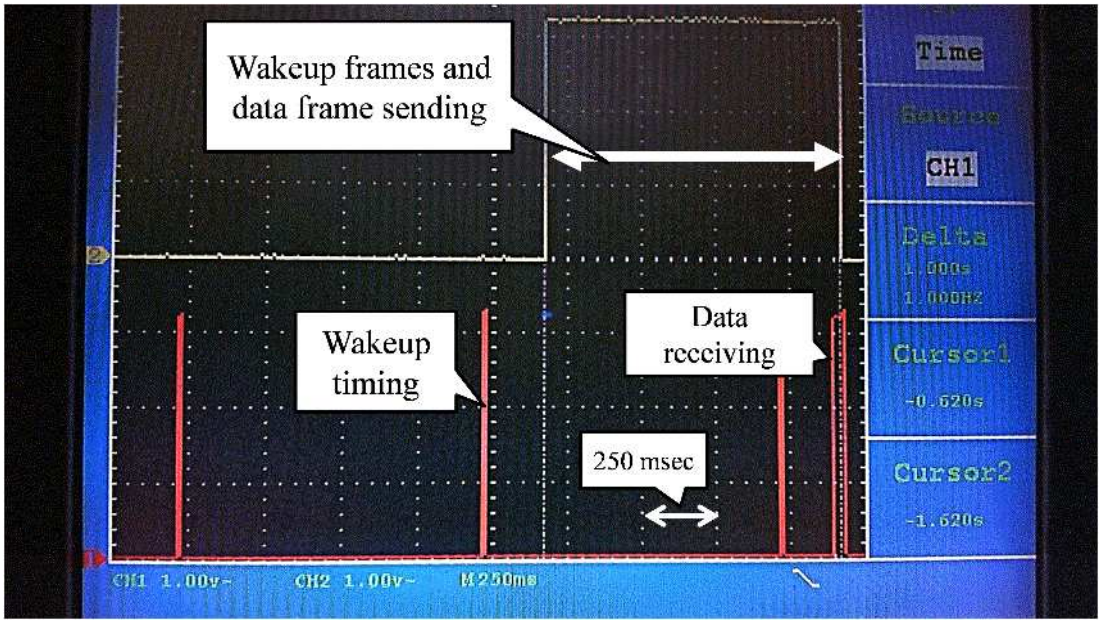

図 13 CSL 非同期通信

IEEE 802.15.4 の CSMA/CA は、データ送信前にランダム時間待機し、その後キャリアセンスを一定時間実施して送信可能かどうかを判断する。この方式を non-persisitent CSMA と呼ぶ。アクセス方式として CSMA を利用した通信は、トラフィック量が増えるとスループットが下がることが知られている。non-persisitent CSMA 方式はキャリアセンス時に送信可能であればすぐパケッ

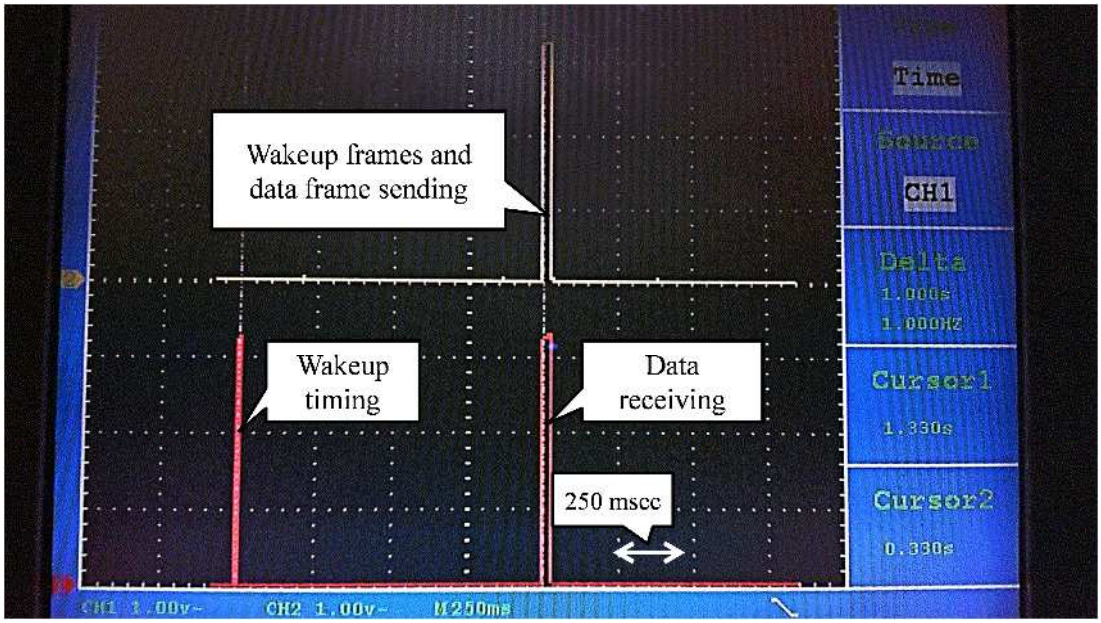

図 15 CSL 同期通信

トを送信する 1-persistent CSMA 方式に比べ、端末毎に待ち時間が異なるので、トラフィック量が高い場合でもパケット衝突確率が少なくスループットが下がりにくいという利点がある[51]。しかし、NES-MAC がターゲットとしているアプリケーションは通信頻度が 1 時間に 1 回程度と、トラフィック量が非常に低く、non-persisitent CSMA と 1-persistent CSMA にスループットの観点から違いはない。よって、NES-MAC では CSMA/CA 時のランダムバックオフを 0 秒に設定し、実質 1-persistent CSMA として動作させる。このことにより、送信側は同期のための計算が容易になる。

3.4.2.2 測定結果

実験はセンサノード 2 台を RS-232C 経由で操作することで実施した。実験時の写真を図 14 に示す。送信タイミング・受信タイミングの測定は、各ノードから GPIO 信号を引き出してオシロスコープで観察した。

図 13 は CSL 非同期通信、図 14 は CSL 同期通信時のオシロスコープの出力である。CSL 同期通信時には Wakeup フレーム送信期間が短くなっている(1 秒から 20 msec)にもかかわらず正しく通信ができていることがわかる。

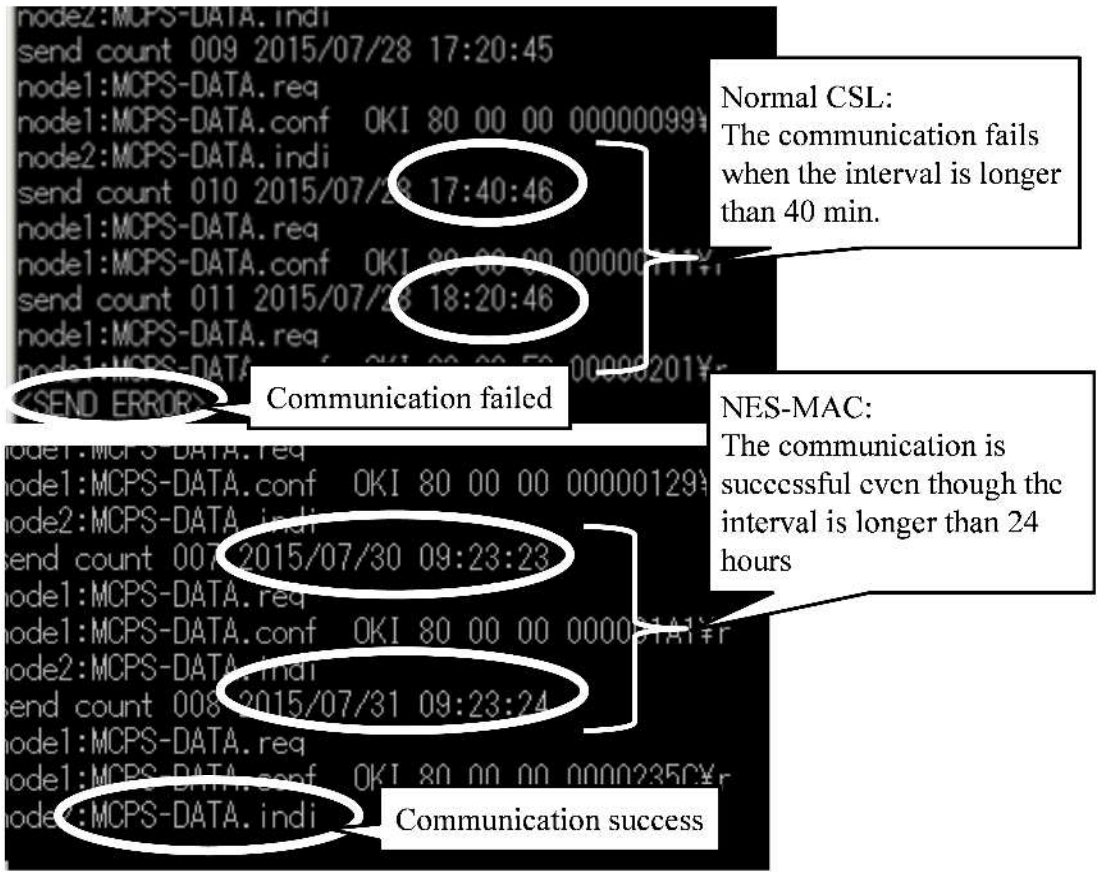

図 16 通信間隔が内外時の動作比較ログ

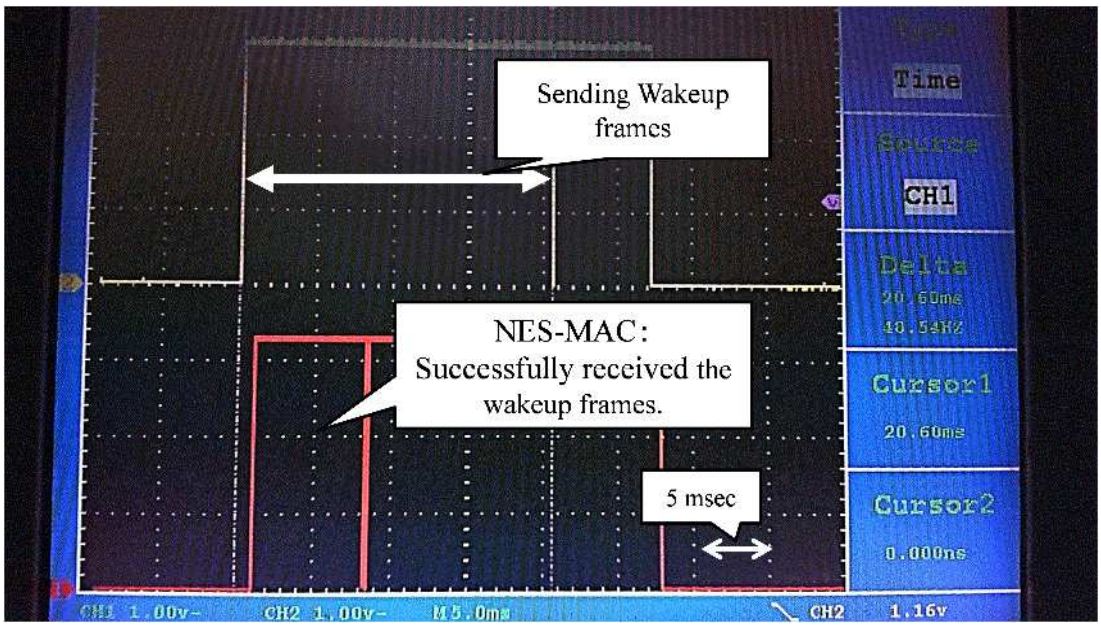

図 17 NES-MAC：通信間隔が 30 分時の動作。通信成功

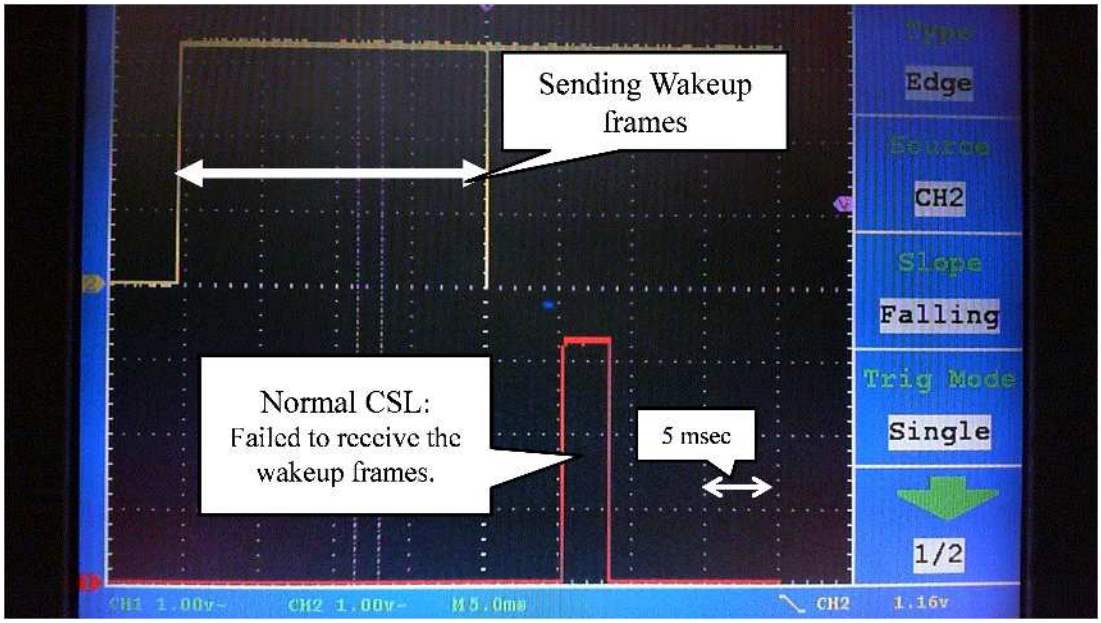

図 18 通常の CSL：通信間隔が 30 分時動作。通信失敗

通常の CSL と NES-MAC 使って、データ送信間隔をアプリケーション要求(UCoMS の要求は 1 時間)に比べて十分長くとって通信をした際の結果を図 16 に示す。上が通常の CSL、下が NES-MAC のログである。上の通常の CSL では通信間隔が 40 分以上開くと通信失敗となるが、NES-MAC の場合は通信間隔が 24 時間開いても通信を継続できることがわかる。

図 17 および図 18 はデータ送信期間を 30 分開けた際の通常の CSL と NES-MAC のオシロスコープの出力である。通常の CSL 時にデータ送信を失敗する理由が、Wakeup フレーム送信開始期間がずれているために受信側の間欠受信タイミングと合わなくなっていることが原因であることがわかる。理論上、30 ppm の CXO を使っていると 30 分で最大約 100 msec 同期がずれるが、図 17 では 15 msec 程度の同期ずれが発生している事がわかる。
クロック補正を実施する NES-MAC では通信は成功している。

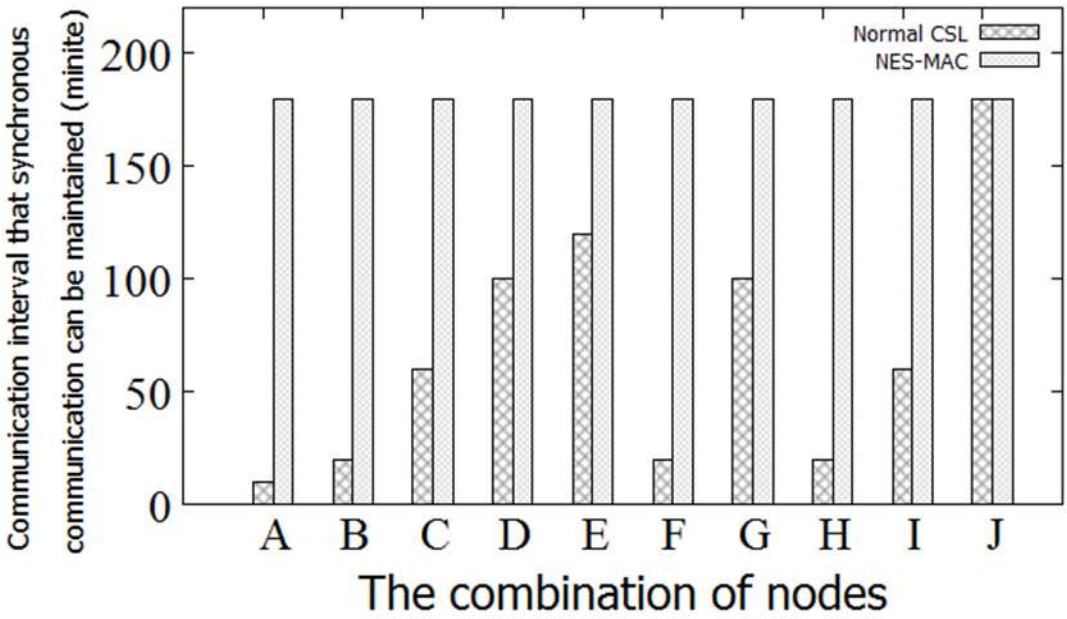

図 19 CSL と NES-MAC の同期を維持する能力の比較 : 通信間隔が 3 時間であっても NES-MAC ノードを同期させることが可能

NES-MAC を用いた場合の性能向上を明らかにするため、約 100 ノードからランダムに 10 組(A–J)のノードを選択し、同期維持性能を測定した。図 19 にその結果を示す。温度変化がほとんどない部屋(約 28±1℃)で測定を行った。通常の CSL の場合、CSL 同期通信が維持され得るデータ通信間隔は、ノードの組み合わせに応じて、約 20 分から約 3 時間まで変化する。これに対して、NES-MAC の場合、データ通信間隔が 3 時間以上であっても、全ての組み合わせにおいて CSL 同期通信を維持できることを確認することができる。

CXO のクロック特性は温度によって変化する。温度変化に対する NES-MAC の許容誤差を明らかにするために、大きな温度変化を伴う通信実験を行った。結果を図 20 に示す。図 20 において、横軸は通信の回数を示し、縦軸は通信中の温度を表している。実験システムを温度変化の大きい窓に設置し、NES-MAC を用いて 1 時間おきに 60 時間通信を行った。本実験では、通常の CSL を用いてデータ送信間隔が 20 分以上になると、CSL 同期通信を維持できないノードの組み合わせを使用した。この実験では、1 時間の通信間隔を UCoMS の要求仕様から設定した。通信中の温度は MH920-Node-232 内蔵の温度測定機能を用いて測定した。実験では、1 時間に 1 回 CSL 同期通信を実施したが、すべての通信で通信が成功し、CSL 同期通信を維持できた。測定期間中、1 時間当たり最大 4℃の温度変化があったが、CSL 同期通信は問題なく維持できたことがわかる。

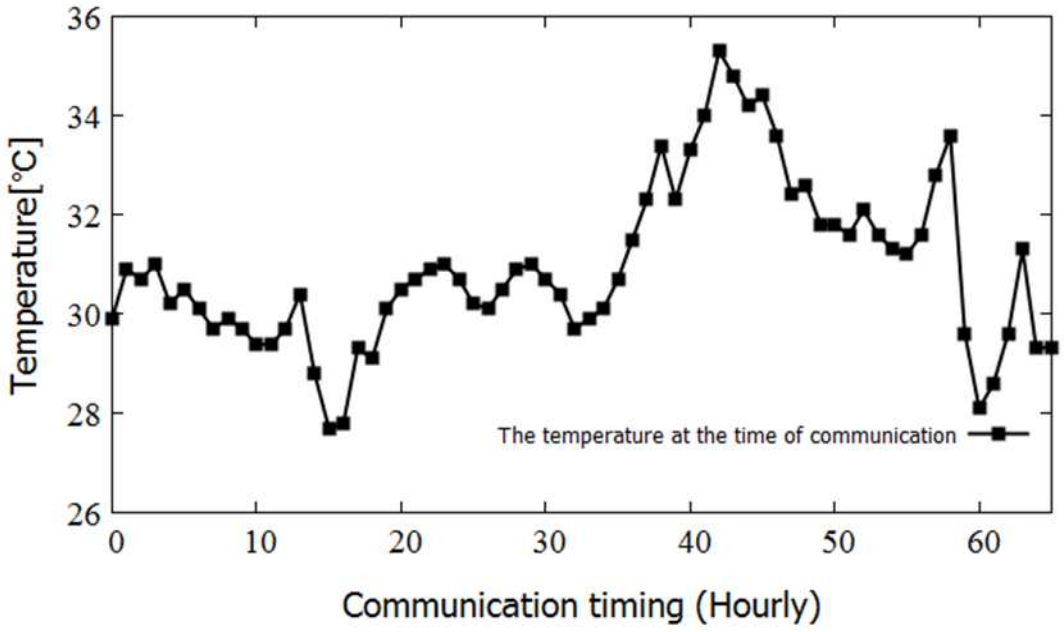

図 20 屋外での NES-MAC の通信性能 : NES-MAC を利用しているので、毎回通信が成功。 1 時間あたり最高 4℃の温度変化があっても同期維持を確認

3.4.2.3 測定結果に対する考察

実証実験の結果、3.2.4.1 で立てた「1 時間に 1 回の通信で CSL 同期通信が維持できれば、現実的なサイズの電池を利用して 10 年間連続動作可能なインフラ監視システムを構築できる」という仮説が現実的であることが示された。NES-MAC を使えば通信間隔が開いても CSL 同期通信を維持できるので、同じ容量の電池を使って長期間連続動作ができる。アプリケーションによってはデータ通信間隔が一定ではない場合もある。NES-MAC を利用すればフレーム送信間隔が一時的に 10 分以上になったとしても CSL 同期を維持できる。

図 6 のグラフを見ると CSL 非同期通信や RIT を使った場合は通信回数が増えるにしたがって 10 年間動作するのに必要な電池容量が大きく増加していく。しかし、NES-MAC を採用して CSL 同期通信が使える場合、週 600 回程度までの通信頻度であれば 10 年間動作するのに必要な電池容量は 1800 ～1900 mAh 程度であまり変化がない。これは 10 年間動作する観点で消費電力を見た場合、週 600 回程度の通信では CSL 同期通信を使ったフレーム送信時の消費電力は、その他の消費電力（間欠受信時、スリープ時）に比べて十分小さいからである。

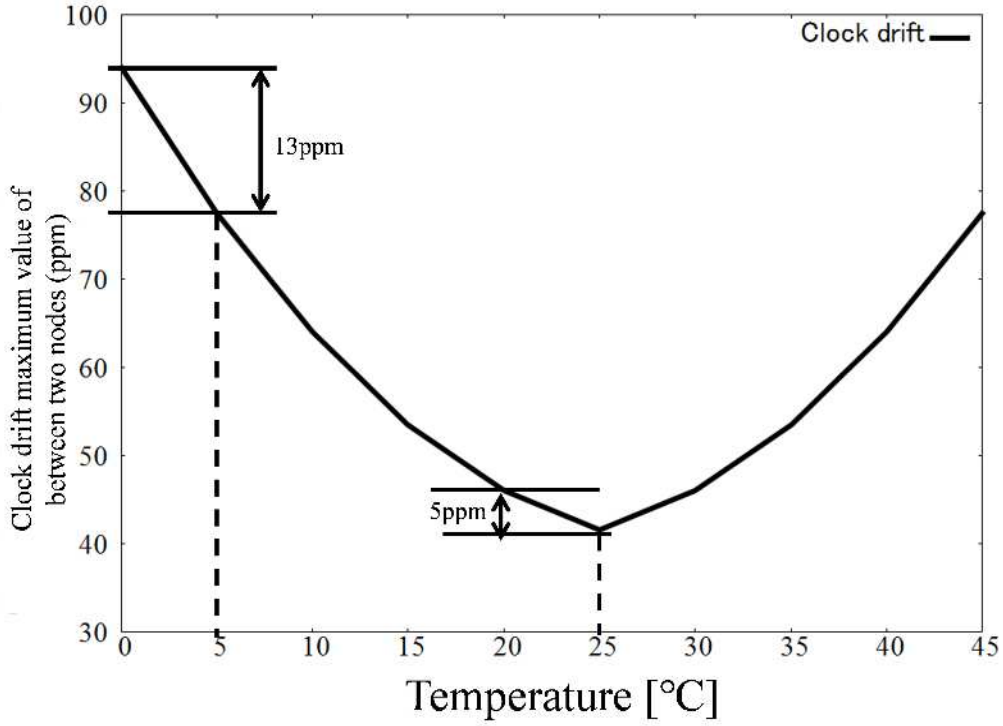

図 21 2 つの異なるノードに組み込まれた CXO 間の最大クロックドリフト値の温度依存性：温度を 5℃変化させたときのクロック誤差は、そのときの温度に依存。例えば、周辺温度が 5°C のときは 13 ppm の変化だが、25℃の場合は 5 ppm 変化

例えば週 600 回程度通信をする場合、CSL 同期通信を利用すると 10 年間でフレーム送信に必要な電池容量は 99 mAh である。これは 10 年間動作するために必要な電池容量である 1895 mAh の 5%程度である。このことから、NES-MAC を利用すると、高々数倍オーダでの通信頻度の変化は、電池容量の観点からは十分許容できるといえる。通信頻度の変化への許容度が高いとシステム設計や構築が容易になるのでシステム導入のための初期コストを低減できる。

また、近年、センサデバイスの消費電力は小さくなっている。例えばテキサス･インスツルメンツ社の TMP102 という温度センサ[52]の駆動電流は 10 μA、アナログ・デバイス社の ADXL372 という加速度センサ[53]の駆動電流 22 μA である。これらの駆動電流は無線部分の駆動電流である 30～50 mA に比べて非常に小さい。よって、無線機能を持つセンサの消費電力は、無線部分の消費電力が支配的であることが多い。この事により、無線部分の動作率を下げて省電力化を図る NES-MAC は、バッテリ駆動の無線を使った IoT センサの実用化に寄与する。

図 19 の実験から、温度変化がほとんどない(1 時間当たりの温度変化が±1 度

未満)屋内で利用する場合、NES-MAC はノードの組み合わせに関係なく、少なくともデータ通信間隔が 3 時間(週当たりの通信回数：56 回)空いた場合にでも同期通信を維持できることが明らかになった。このことより、NES-MAC は例えば UCoMS のような屋内向けのインフラ監視システムに利用できる。

図 20 の実験から温度変化が大きい(1 時間当たりの温度変化が最大で±4 度程度)屋外で利用する場合でも、NES-MAC を使えば通信頻度が 1 時間に 1 回程度(週当たりの通信回数：168 回)で同期通信が維持できることが明らかになった。

現在の NES-MAC の実装では、CSL 同期通信時の Wakeup フレーム送信期間は 20 msec である。よって、1 時間のクロック誤差変化が 10 msec になると同期が外れる。このことにより、NES-MAC では、1 時間当たりのクロック誤差が 10 msec / 1 時間 = 約 2.8 ppm 変化すると同期が外れるといえる。

図 21 は MH920-Node-232 が使用している CXO (リバーエレテック TFX-03)のデータシート[38]から算出した、温度変化とノードに内蔵されたクロック間の最大誤差変化の関係図である。ノードに内蔵された CXO 間の最大誤差量は温度に依存する。例えば 25℃近傍では温度が 5 度変化しても最大誤差は 5 ppm 程度である。一方 5℃近傍では温度が 5 度変化すると最大誤差は 13 ppm に達する。現在の NES-MAC の実装は UCoMS [14]向けに屋内での利用を想定して実装されている。NES-MAC を屋外で利用する場合、利用環境の温度変化に対応するために CSL 同期通信時の Wakeup フレーム送信期間を増やす必要がある。たとえば利用環境の温度変化が 0℃～40℃であり、1 時間当たりの温度の変化量が最大 5 度とすると、クロック誤差の変化量は図 12 より最大 13 ppm である。このことにより、時刻同期をした後に 5 度温度変化があると 1 時間当たりのクロック誤差変化が最大 46.8 msec になることとなる。この場合、NES-MAC の CSL 同期通信時の Wakeup 送信期間を 46.8 × 2 = 93.6 msec に設定しておくと同期はずれが発生しないと考えられる。従って、NES-MAC は設定変更によってインフラ監視目的で屋外での利用も可能である。

第 4 章　NES-SOURCE：制御向け低遅延かつ軽量ネットワークプロトコル

4.1　イントロダクション

2.2 節で論じたとおり、遅延保証が求められる制御無線ネットワーク(Wireless Control Network。以下、WCN)はアクセス方式としてパケット衝突が発生しない TDMA 方式を採用する場合が多い。これは無線通信での遅延の原因はパケットロスであり、パケットロスの原因はパケット衝突発生が原因だと考えられているからである。しかし、TDMA は 2.2 節で論じたとおり、非常に扱いづらいという課題がある。アクセス方式として CSMA/CA 方式を採用すると、システムはコンパクトかつ容易に構築可能になる。

そこで、2.2.2 項で上げた WCN の通信トラフィック下でどの程度パケット衝突が発生するかを簡単なシミュレーションを使って測定した。その結果、WCN の通信トラフィックは、アクセス方式として TDMA 方式を利用する必要があるほど高いものではなく、CSMA/CA 方式で十分対処可能であることが分かった。

2.2 節で述べたとおり、パケットロスの原因はパケット衝突の発生と通信環境の変化である。WCN は工場など、人や物が出入りするような箇所での利用が想定される。こういった環境では、通信経路に人や物など通信遮蔽物が入り通信環境が変化し、パケットロスが発生することが多い。遮蔽物によるパケットロスが発生した場合、同じ通信路での通信は難しく、通信経路の変更で対応する必要がある。2.2.1 項で論じたとおり、アクセス方式として TDMA 方式を利用した場合は通信路の多重化が容易である。しかし、アクセス方式として CSMA/CA を利用している場合、通信路を多重化すると ACK 通信とデータ通信が干渉するためにパケット衝突発生確率があがる。よって、遮蔽物によるパケットロスが発生した場合、それをすばやく検知して通信経路を変更する方式が必要である。

本章では、コンパクトで実用的な制御無線ネットワーク用のプロトコルスタックである NES-SOURCE を提案する。NES-SOURCE の実装はコンパクトかつ理解や変更、メンテナンスが容易である。具体的には、コードサイズで比較

すると一般的な近距離無線用通信プロトコルである ZigBee PRO の約 4 分の 1 であり、TDMA ベースの WCN プロトコルの約 2 分の 1 である。NES-SOURCE プロトコルスタックを構成する関数のソースコードの複雑さを示す McCabe 数は概ね 10 以下であり、これは他のプロトコルスタックよりも少ない。これは、NES-SOURCE スタックのカスタマイズしやすさおよび調整のしやすさを示している。

TDMA ベースの WCN は障害回避のために通信の経路を複数持つ。経路を複数持つことで、ある経路で通信障害が発生したときもパケットロスが発生しない。一方、NES-SOURCE は、通信障害が検出されたときに通信経路が迅速に切り替わる、高速な通信経路切り替え機能を持つ。NES-SOURCE はこの機能により、通信経路を複数持たないにもかかわらず低遅延と低 PER を実現する。

単純な固定パスプロトコルと比較して、NES-SOURCE には、通信経路を高速に切り替えるために以下の機能を持つ。

(1) バックオフ時間の短縮

バックオフ時間が短くなるとパケットの遅延が低くなることは知られている[54]。WCN で使用されるセンサデータおよびコマンドは、高々数十バイトである。そこで、フレームサイズを IEEE 802.15.4d [39]の最大フレームサイズ値と同じ 128 byte に制限することにより、CSMA / CA によるランダムバックオフ時間を、IEEE 802.15.4g の最大バックオフ時間である 15 msec から最大 2.8 ms に低減した。

(2) ACK 省略

通信経路が 1 HOP の場合、NES-SOURCE が MAC レイヤのパケット損失を検出すると、NES-SOURCE はネットワークレイヤの ACK タイムアウトを待たずに通信経路を切り替えてフレームを再送信する。その結果、1 HOP 時の通信時間は最大で約 62%低減できた。

以上の 2 つの特徴により、NES-SOURCE の通信遅延は正規ルートが 1 HOP、バックアップルートが 2 HOP という通信路で最悪遅延が 60 msec 程度まで短縮

できた。また、人の出入りで通信路へランダムに通信障害物が発生するような環境にて、NES-SOURCE の経路変更機能が有効に働くことで PER や通信遅延が改善することを実証実験で確認した。さらに、通信路への障害物侵入確率およびパケット衝突発生による PER と、NES-SOURCE の PER の関係を明らかにした。このことにより、NES-SOURCE の経路変更方法を、システム設計の段階で最適値に設定できる。

NES-SOURCE は産業界に受け入れられやすい、以下のような特徴を持つ。

(1) 標準準拠

NES-SOURCE は IEEE 802.15.4d に準拠している。IEEE 802.15.4d は、サブギガ帯域(920 MHz 帯域)を使った CSMA/CA ベースの MAC プロトコルである。IEEE 802.15.4g に比べ小型のセンサデータを収集することに向いたプロトコルである。

(2) ハードウェアの調達が容易

NES-SOURCE が準拠している IEEE 802.15.4d は、IEEE 802.15.4g に準拠した RF 用 LSI を利用して実装できる。よって、世の中一般に出回っている IEEE 802.15.4g 準拠の RF 用 LSI を利用でき、低コストでシステム開発ができる。また、TDMA を採用していないため、同期精度を上げるために製造時に特別な施策等は不要である。高価なクロック素子を採用する必要もない。

(3) 低保守・運用コスト

NES-SOURCE はコードサイズも小さく、コード辞退の複雑さも低く実装されているので、通信プロトコルスタックへの機能の追加やメンテナンスが容易である。また、アクセス方式として TDMA を採用していないため、システムへの新しいハードウェアの追加も専門知識無しに実施できる。

4.2 関連研究および問題点

4.2.1 標準技術

WCN へ応用可能な通信プロトコルはいくつか規定、標準化されている。近距離向けの無線通信プロトコルとして最も標準的なものとして、IEEE 802.15.4 [4]がある。IEEE 802.15.4 は 2.4 GHz 帯域を利用した近距離無線通信のための MAC プロトコルであり、ZigBee PRO や WirelessHART など、様々な通信プロトコルの MAC 層に採用されている。IEEE 802.15.4 はアクセス方式として CSMA/CA を使うノンビーコンモードと、アクセス方式として TDMA と CSMA/CA を組み合わせて使う、ビーコンモードがある。2.2 節で述べたとおり、アクセス方式として TDMA 方式を利用した場合、パケット衝突が発生しないという利点はあるが、自身に割り当てられたスロット以外では通信できないので、急を要する通信要求には対応できないという欠点がある。IEEE 802.15.4 のビーコンモードはこういった課題に対応するため、CSMA/CA 方式で通信をするためのスロットである ContentionAccessPeriod と、TDMA 方式で通信をするためのスロットである ContentionFreePeriod を持つ構成となっている。ロスが許されないパケットは ContentionFreePeriod で通信をする。ContentionFreePeriod は完全な TDMA 通信なのでパケット衝突によるパケットロスは発生しない。遅延要求が厳しいパケットは ContentionFreePeriod まで待たずに CSMA/CA を使って ContentionAccessPeriod で送信される。

ビーコンモードはよく考えられた方式であるが実装や運用が難しく、商用ベースの実装は無い。

Time Slotted Channel Hopping (TSCH)は IEEE 802.15.4 の拡張機能として IEEE 802.15.4e [26]で標準化された TDMA ベースの MAC をプロトコルである。TSCH はタイムスロットごとに通信チャネルを変更するので雑音に強い。この特徴により WirelessHART の MAC 層に採用されている。しかし、TSCH は運用のために高い同期精度が必要である。例えばリニアテクロノジ社の WirelessHART モジュールは、製造時にクロックの誤差を測定して誤差をハードウェアに埋め込むことで高い同期精度を実現している。こういったハードウェアは汎用品で作ることができず調達コストが高くなる。

WirelessHART は、実際に運用されている代表的な WCN 向けのネットワークプロトコルである。TSCH を使用して、冗長通信パスを組み込むことによって通信の信頼性を向上させている。しかし、WirelessHART は、基地局側でネットワーク環境を常に監視し、通信経路を構築しノードへ配信している。よって、何らかの理由でノードと基地局間の通信が途切れた場合、システムは破綻する。更に、WirelessHART は TDMA を使用しており、50～100 ms 程度の遅延を保証することは難しい[22]。

免許不要で利用できる無線通信の周波数帯域は、430 MHz 帯域、920 MHz 帯域、2.4 GHz 帯域がある。この内 430 MHz 帯域は、通信距離は長いが通信速度が 4.8 kbps と低いため制御無線には利用しにくい。制御無線ネットワークで利用する通信帯域として 920 MHz 帯域が注目されている。920 MHz 帯域は IEEE 802.15.4 で利用する 2.4 GHz 帯域に比べて長距離通信が可能である。また、回り込み性能も高い。更に、2.4 GHz 帯域は ISM (Industrial, Scientific, Medical)バンドして様々な用途に使われるので 920 MHz に比べて雑音が多く、パケットロスが発生しやすい。日本では IEEE 802.15.4g [29]が 920 MHz 帯域で使える通信規格である。IEEE 802.15.4g は、通信レートこそ 100 kbps と IEEE 802.15.4 の 250 kbps に比べて遅いが、通信距離は 20 mW 出力で見通し 1 km 以上になる。これは通信距離が見通しで高々250 m 程度である IEEE 802.15.4 に比べて長い[35]。通信距離の長さはデータのスループットにも影響する。例えば見通しで 1 km 先の基地局に対して 50 byte のデータを送信する場合、920 MHz 帯域を使うと 1 HOP で届くので 50byte÷100 kbps×1 HOP=4 msec で通信できるが、2.4 GHz 帯域を使うと 4 HOP かかるので 50 byte÷250 kbps×4 HOP=6.4 msec かかる。ちなみに、後者の通信時間は経由ノードのフレーム転送にかかる時間や通信失敗時の再送にかかる時間を無視して計算したため、実際には両者の差は更に大きくなる。以上のことから NES-SOUCE の PHY 層としては 920 MHz 帯域が最適であると言える。ただし、920 MHz 帯域を利用する通信規格である IEEE 802.15.4g の最大フレームサイズは 2048 byte になる。このため、CSMA/CA 通信時のバックオフ時間は最大 15 msec 程度になる。このバックオフ時間は 50～100 msec 程度の通信遅延を要求する制御無線ネット

ワークには大きすぎるので使いにくい。

4.2.2 既存研究

WCN のための通信プロトコルは色々と研究されている。WCN プロトコルはアクセス方式として TDMA を採用した方式と CSMA/CA を採用した方式で大別できる。

アクセス方式として TDMA を利用した WCN の研究は WirelessHART、PEDAMACS [40]、 GinMAC [23]がある。WirlessHART はサーバがルーティングやチャネル割り当てをしていた。よって、サーバからの通信が途絶えると通信ができなくなるという課題がある[24]。

PEDAMACS はサーバの送信出力を上げることによって、サーバから各ノードへの通信を 1 HOP で実現することによりこの問題を解決している。しかし、PEDAMACS を WCN の通信プロトコルとしてとして利用する場合、通信環境によっては出力を上げてもサーバからの 1 HOP 通信が不可能である場合がある。例えば UCoMS [14]では出力を上げても 1 HOP 通信が不可能な、配管が複雑に入り組んだ環境での利用を前提としている。

GinMAC はこの問題をノードへルーティング情報を事前に入力することで回避している。しかし、GinMAC は TDMA を利用しているので、4.1 節であげた TDMA 方式の課題は残る。

アクセス方式として CSMA/CA を利用した WCN の研究は MMSPEED [41]や Dwarf [42]がある。

MMSPEED はデータ流の属性による QoS 制御を採用することで遅延保証に対応している。しかし、この方式はネットワーク規模が大きくネットワーク上に流れるデータ量が大きい場合に効果を発揮する。例えば MMSPEED の対象としているネットワークの規模は 150 台程度であり、制御無線ネットワークが主に対象としているネットワーク規模である 25 台[24]程度と比べて大きい。

Dwarf はフラッディングベースのユニキャストを利用することで通信の信頼性を上げる方式である。この方式も設置密度やネットワーク規模が大きい場合に効果を発揮する。しかし、制御無線ネットワークが対象とする規模では効果

を発揮しにくい。また、フラッディングベースなのでパケット衝突が多発し、通信遅延が発生する可能性が高い。

4.3　制御無線ネットワークに対する要求条件の整理

4.3.1 パケットロスを低減するために必要な機能

2.2.2 項で上げた制御無線ネットワークへの要求条件を満たすためには、通信遅延を減らす必要がある。無線通信での数 msec オーダ以上の遅延はパケットロスによる再送がその原因である。よって、パケットロスを減らすことは通信遅延を減らすために重要である。

2.2 節で述べたとおり、パケットロスの原因は、通信環境の変化とパケット衝突の発生である。NES-SOURCE は、工場などの人や機器が出入りする環境での利用を想定している。よって、こういった状況では、通信路が人や物によって一時的に遮断され、通信環境が変わりパケットロスが発生することは想定できる。

ZigBee PRO のような近距離通信向けネットワークプロトコルの場合、こういった課題は、各ノードの自律的な経路変更で対応する。経路を自律的に選ばない WirelessHART のような制御無線ネットワークプロトコルでは、こういった課題に対してフレーム送信時に 2 つの経路に対してフレーム送信する「通信路の 2 重化」によって対応している。

2.2.1 項で論じたとおり、WirelessHART のようにアクセス方式として TDMA を採用している場合には通信路を 2 重化してもパケット衝突は発生しない。しかし、アクセス方式として CSMA/CA 方式を利用する場合には通信路を 2 重化するとパケット衝突が発生しやすい。これは同一送信先にほぼ同じタイミングでデータを送信するからである。よって、CSMA/CA 方式を利用した WCN プロトコルスタックを検討する場合、ある経路の通信が失敗した場合に短時間でそれを検知して通信経路を高速に切り替える方式を検討する必要がある。

4.3.2　予備実験：制御無線環境下でのパケット衝突発生確率

表 7 シミュレーション設定値

aUnitBackoffPeriod [8]	200 μsec
aCCATime [8]	100 μsec
macMinBE [8]	3
macMaxBE [8]	4
macMaxCSMABackoffs [8]	5
The number of peripheral nodes	25
Frame send period	3.5 msec
Frame send interval	1 sec
Measurement time	30000 sec×50

```
/* Make node send schedule */
 node_send_schedule(); … (1)
 /* Peripheral nodes send dchedule */
 peripheral_node_send_schedule(node_num = 25) … (2)

 /* The Time is 0.1 msec increments. */
 for(Time = 0;Time<MAX_TIME;Time++){ … (3)
  /* change peripheral node status */
  change_peripheral_node_status(); … (4)
  /* change node status */
  if(SOMEONE_SEND == peripheral_node_status() && SEND == node_status ){ … (5)
   SEND_MISS_COUNT++;
   change_node_status(SNED_MISS) … (6)
  }
```

図 22 パケット衝突発生シミュレータの擬似コード

2.2.2 項で論じた WCN のトラフィック下でどの程度のパケット衝突が発生するか、簡単なシミュレータを用いて測定した。シミュレータの条件を表 7 に、またシミュレータの擬似コードを図 22 に、各ノードの状態遷移を図 23 に示す。CSMA/CA 方式は一般に、「送信があればまず、ランダム時間連続でキャリアセ

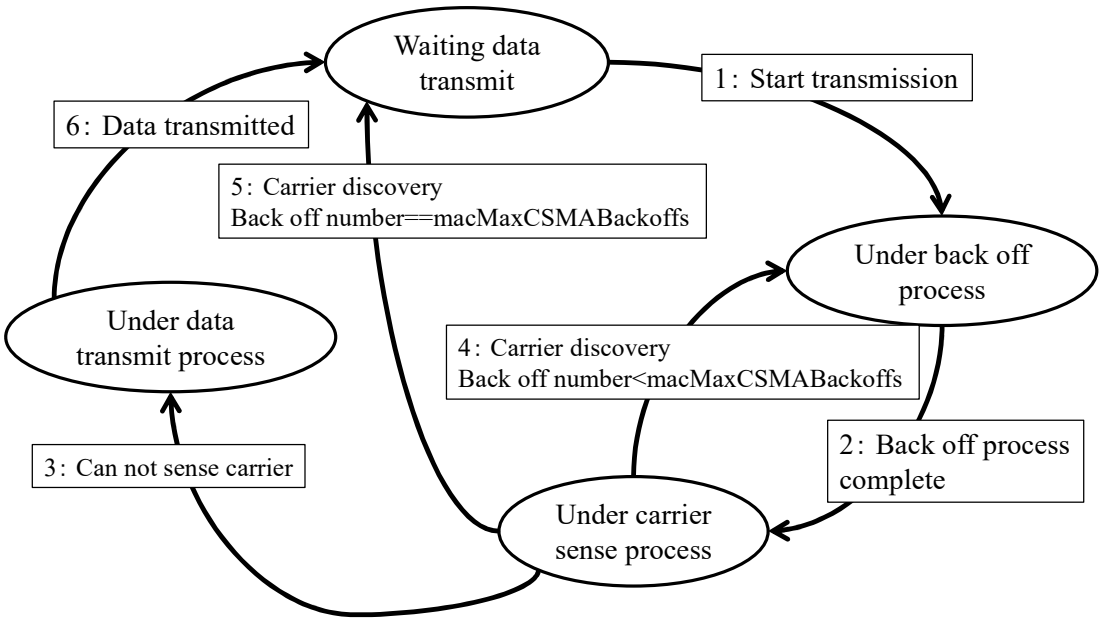

図 23 シミュレーションでの送信ノードの状態遷移図

ンスをし、通信チャネルが空いていたらフレームを送信する」という方式である。IEEE802.11 ではこういった通常の CSMA/CA 方式の動作をする。しかし、IEEE 802.15.4 の CSMA/CA 方式のアルゴリズムは IEEE 802.11 の方式とは違い、non-persistent CSMA 方式を利用している。具体的には、送信要求後のランダム時間待ちの際には連続してチャネル観測をしない。ランダム時間待った後、一定時間(IEEE 802.15.4g の場合、130 μsec)キャリアセンスをし、通信チャネルが空いていたらフレーム送信をする[4]。この方法は IEEE 802.11 の方式に比べてパケット衝突発生率は高くなるが、消費電力は下がる。IEEE 802.15.4 は元々IoT 通信のようなトラフィックが高くない通信を対象としていたため、このような仕様になっている。

シミュレータではまず、ノードおよび周辺ノードの送信タイミングスケジュールを作成する(図 22(1)および(2))。送信タイミングはシミュレータ内の 1 秒毎±500 msec に 0.1 msec 単位で完全にランダムに発生するようにした。

シミュレーションは 30000 sec (図 22(3))の間、事前に決められたタイミングで状態を遷移させる(図 22(4))。各ノードは「送信待ち」「バックオフ中」「キャリアセンス中」「送信中」の 5 状態がある。各ノードは図 23 の状態遷移図に従って状態を遷移させる。

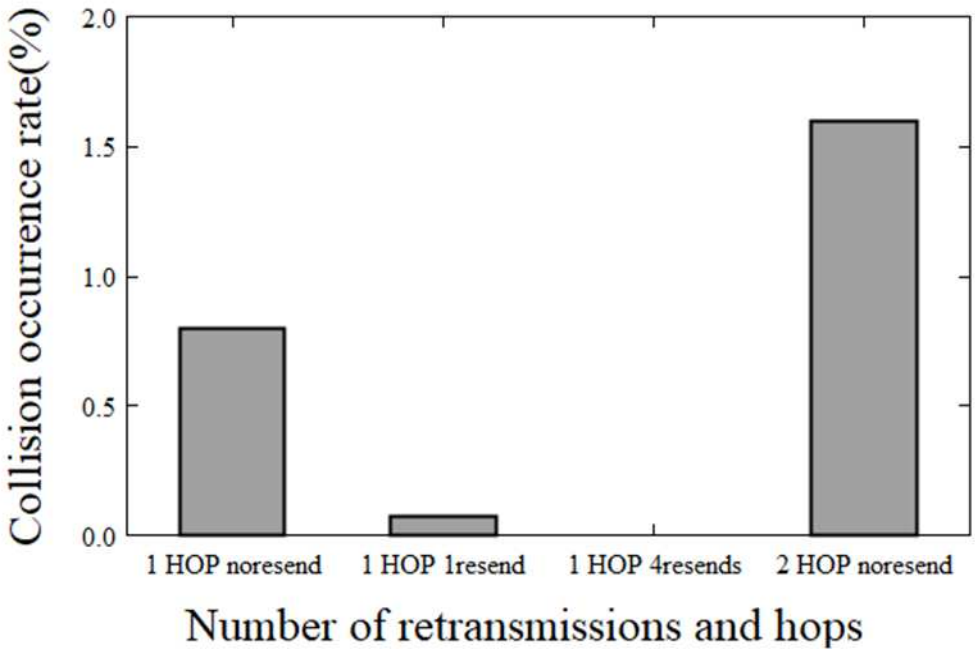

図 24 予備実験結果：制御無線環境下でのパケット衝突発生率

ノードは普段送信待ち状態である。「1：Start transmission」時刻になれば、バックオフ状態になる。バックオフは以下の式で計算できる。

$$\text{BackoffPerio} = aUnitBackoffPeriod \times 2^{rand(\min(macMinBE+\text{バックオフ回数、}\ macMaxBE)+1)} \quad (19)$$

「2：Back off process complete」後、aCCATime の期間、キャリアセンスをする。キャリアセンスの結果、「3：Cannot sense carrier」場合は、フレーム送信をし、「6：Data transmitted」後、送信待ち状態へ戻る。キャリアセンス中に「4：Carrier discovery Back off number<macMaxCSMABackoffs」場合は、バックオフを再開する。「5：Carrier discovery Back off number == macMaxCSMABackoffs」の場合は、通信失敗として送信待ち状態になる。送信ノードが SEND 状態(送信状態)の場合に周辺ノードのうち 1 つでも送信状態であれば(図 22(5))、送信失敗(図 22(6))となる。

シミュレーションの結果、再送なしの 1 HOP の場合の PER は 0.8%、再送なしの 2 HOP の場合の PER は 1.6%、1 回再送の 1 HOP の PER は 0.07%、4 回再送の 1 HOP の場合の PER は 0%であった。特に、再送を実施するとパケット衝突発生によるパケットロスの確率は非常に低くなることがわかる(図 24)。この簡易シミュレーション結果によりパケット衝突発生によるパケットロスの確

率は 2.2.2 項であげた WCN の要求に比べて低いことがわかる。

4.4　NES-SOURCE のデザイン

我々は 2.2.2 項であげた制御無線ネットワーク条件を満たす、コンパクトな制御無線ネットワーク向けプロトコルスタックである NES-SOURCE を設計・開発した。NES-SOURCE の特徴を以下に示す。

4.4.1　PHY/MAC 層

NES-SOURCE の PHY・MAC 層は IEEE 802.15.4g をベースとしている。利用周波数帯域は 920 MHz 帯域で、変調方式は GFSK、出力は 20 mW である。

一般論として、通信距離が長いとマルチホップネットワークのホップ数が少なくなり、結果として通信遅延も小さくなる。4.2.1 項で示したとおり、IEEE 802.15.4g の見通しでの通信距離は 2.4GHz を使う IEEE 802.15.4 の 4 倍であり、同じ距離を通信する際の HOP 数も少なくなり、通信遅延が小さくなる確率が高い。また、920 MHz 帯域は 2.4 GHz 帯域の無線通信方式に比べて回り込み特性がよいので、例えば UCoMS が想定しているような配管が入り組んだ場所での利用にも適している。

送信するフレームサイズが同じである場合に CSMA のバックオフ単位時間を増やしていくと、コリジョン回避確率が増えるのでスループットが上がる。しかし、バックオフ単位時間をある一定の値以上に増やすと、バックオフが遅延に与える影響が、通信遅延に比べて大きくなるのでスループットが下がることが知られている[54]。つまり、バックオフの単位時間はただ単に大きくすればよいというものではなく、パケットサイズに合わせた値を設定する必要がある。

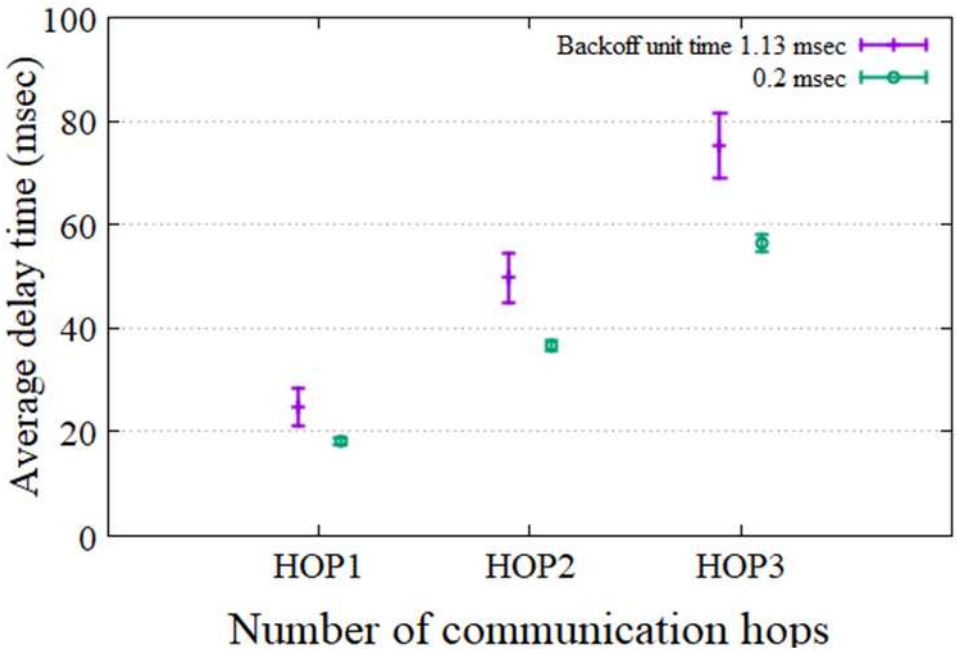

図 25 バックオフの単位時間を変更させた際の通信ホップ数と平均通信遅延時間の関係

IEEE 802.15.4g の MAC 層の CSMA のバックオフの単位時間は 1.13 msec である。これは日本向け 950 MHz 帯域の無線通信規格である IEEE 802.15.4d [39]の CSMA のバックオフの単位時間である 0.2 msec に比べて長い。これは IEEE 802.15.4g の最大フレームサイズが 2048 byte と IEEE 802.15.4d の最大フレームサイズである 128 byte に比べて長いからである。IEEE 802.15.4g のフレームサイズは、近距離無線通信で IPv6 パケットを利用できるように決められた。しかし、WCN では画像や音声データを送受信するわけではないので、2048 byte もの大きなフレームを通信する必要はない。フレームサイズは IEEE 802.15.4d レベルのサイズで十分である。バックオフは通信をおこなう度に実施される。よって、通信遅延の観点からすると短ければ短いほど良い。

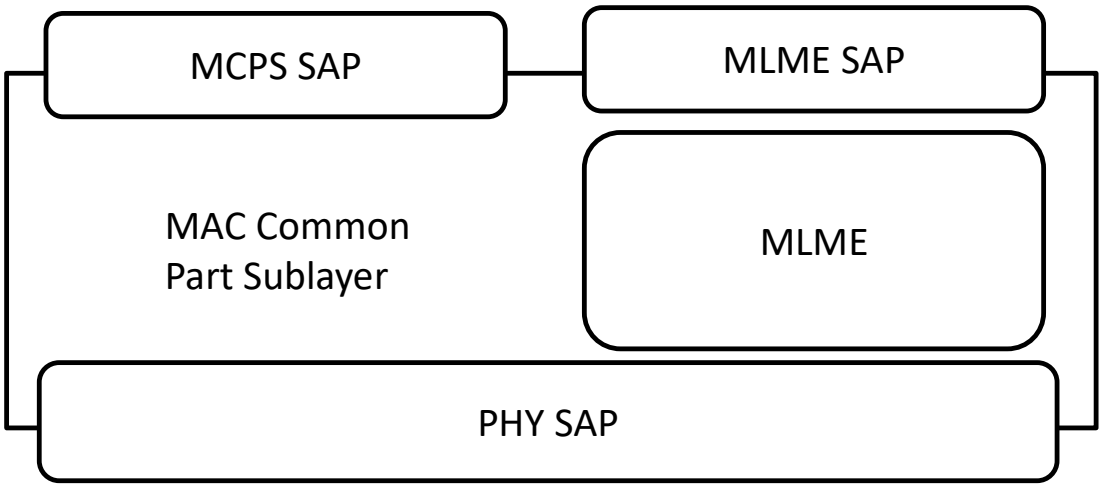

図 26 IEEE802.15.4 のスタック構成図

バックオフの単位時間と通信遅延の関係を図 25 に示す。通信遅延は PHY および MAC の設定を表 7 の条件に設定した上で、NES-SOURCE の NW-ACK 要求有りのデータ通信完了までの時間を測定することで行った。バックオフの単位時間が 1.13 msec の場合、1 HOP、2 HOP、3 HOP の平均通信遅延はそれぞれ 24.7 msec、49.7 msec、75.3 msec であり、その標準偏差はそれぞれ 3.6 msec、4.7 msec、6.3 msec であった。また、バックオフの単位時間が 0.2 msec の場合、平均通信遅延はそれぞれ 18.1 msec、36.6 msec、56.4 msec であり、その標準偏差はそれぞれ 0.6 msec、0.9 msec、1.6 msec であった。平均遅延時間で比較すると、バックオフの単位時間を 0.2 msec にすることで遅延時間を平均 29%改善できることがわかる。制御無線ネットワークでは大きなデータを送受信する必要はなく、フレームサイズの最大値は 128 byte で十分である。よって、NES-SOUCE のランダムバックオフ時間は IEEE 802.15.4d のものを採用することにした。

表 8 NES-SOUCE で使う IEEE 802.15.4 コマンド

MCPS-DATA.request	Data transmission request
MCPS-DATA.confirm	Notification of data transmission result
MCPS-DATA.indication	Receive data notification
MLME-SET.request	MAC layer setting change request
MLME-SET.confirm	MAC layer setting change result
MLME-GET.request	MAC layer setting acquisition request
MLME-GET.confirm	MAC layer setting acquisition result

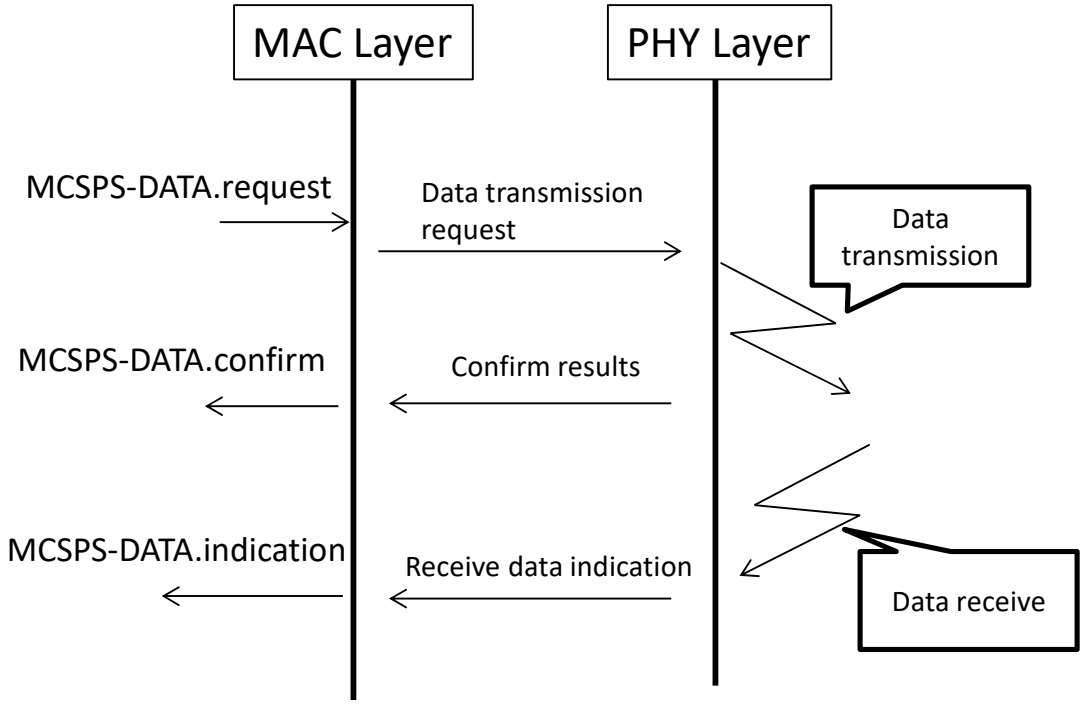

図 27 MCPS-DATA の動作シーケンス

IEEE 802.15.4 は内部構造としてデータフレームのやり取りといった基本的な動作を司る MAC common part sublayer (MCPS)と、IEEE 802.15.4 で規定されたコマンド通信や MAC 層の設定(16 ビットアドレス等)を司る MAC sublayer management entity (MLME)からなる(図 26)。MCPS および MCPS とのインタフェースは Service Access Point (SAP)と呼ばれる。本実装では、上位レイヤから IEEE 802.15.4 準拠の MAC プロトコルスタックを操作する場合、SAP を介して MCPS および MLME へコマンドを発行する形で操作するよう実

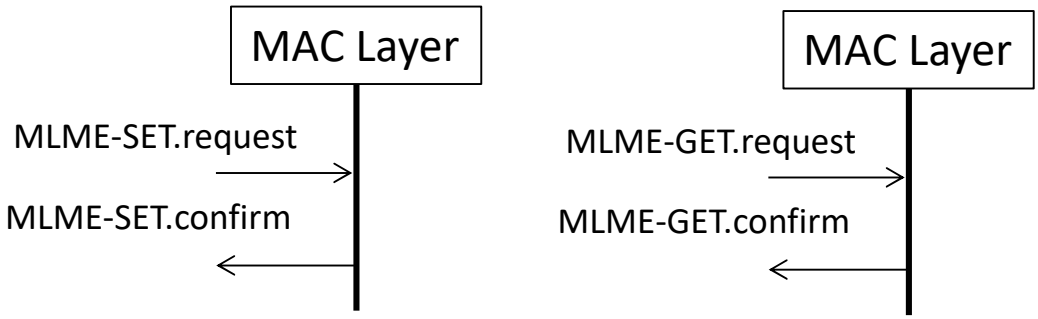

図 28 MLME-SET and MLME-GET の動作シーケンス

装した。

IEEE 802.15.4 の規格では様々なコマンドが用意されているが、NES-SOURCE では表 8 にあげるコマンドのみ利用する。各コマンドにはコマンド固有の引数がある。コマンドを利用する際には、引数を設定してコマンドを利用する。

MCPS-DATA.request、MCPS-DATA.confirm、MCPS-DATA.indication はデータ通信に関するコマンドである(図 27)。MCPS-DATA.request はデータフレームを送信するためのコマンドである。NES-SOURCE が利用する MCPS-DATA.request の引数は、送信先アドレス、送信データ、MAC 層の ACK の有無である。引数として MAC 層の ACK 有りで MCPS-DATA.request コマンドを発行した場合、IEEE 802.15.4 はデータフレーム送信後に ACK がなければ予め決められた回数再送をする。MCPS-DATA.confirm は、IEEE 802.15.4 プロトコルスタックが MCPS-DATA.request の実施結果を上位レイヤへ通知するためのコマンドである。通知する実施結果としては「通信成功」「ACK なし」「キャリア検知により通信失敗」等がある。MCPS-DATA.indication は、IEEE 802.15.4 プロトコルスタックが受信した受信フレーム情報を上位レイヤへ通知するためのコマンドである。ここでいう受信フレーム情報には、少なくとも送信元アドレスやフレームのペイロード(MCPS-DATA.request の引数で「送信データ」と指定したもの)が含まれる。MLME-SET.requestm および MLME-SET.confirm は MAC 層の設定を変更するためのコマンドである(図 28 左)。

引数として設定項目及び設定値を指定して MLME-SET.request コマンド発行すると、設定結果が IEEE802.15.4 スタックから MLME-SET.confirm コマンドという形で返ってくる。MLME-SET.confirm の内容は「設定成功」および「設

2 byte	1 byte	6 byte or 18 byte	*N* byte	2 byte
Frame Control	Data SN	Address Information	Data Payload	FCS

図 29 MAC レイヤのフレーム構成

定失敗」である。MLME-GET.request および MLME-GET.confirm は MAC 層の設定値を取得するためのコマンドである(図 28 右)。

引数として設定値を取得したい設定項目を指定して MLME-GET.request コマンドを発行すると、取得結果が IEEE 802.15.4 スタックから MLME-GET.confirm コマンドという形で返ってくる。MLME-GET.confirm の内容は「MLME-GET.request で指定した設定項目の設定値」もしくは「設定項目エラー(設定項目が見つからない)」である。

表 8 であげたコマンドの内、IEEE 802.15.4 スタックからのコマンドである MLME-SET.confirm および MLME-GET.confirm は MLME-SET.request および MLME-GET.request の実装である関数の戻り値として実装をした。一方、MCPS-DATA.confirm、MCPS-DATA.indication は、IEEE 802.15.4 スタックからのメッセージを、予め用意した MessageQueue で待つという形で実装をした。以下に詳しく説明する。

MLME-SET.confirm および MLME-GET.confirm は、MLME-SET.request および MLME-GET.request 実行直後に発行できる。なぜなら、MLME-SET.request および MLME-GET.request は MAC 層の設定値にアクセスするだけであり、コマンド実施および結果の取得にかかる時間はごく短い(μsec オーダ)。よって、MLME-SET.confirm および MLME-GET.confirm は関数の戻り値として実装をした。一方、MCPS-DATA.request コマンドは、内部でバックオフやキャリアセンス、ACK を使った再送処理をするために、コマンド終了までの時間がかかる(～数十 msec オーダ)。MCPS-DATA.request の結果を通知するコマンドである MCPS-DATA.confirm を、MCPS-DATA.request の実装である関数の戻り値として実装すると、プログラム処理がそこで数十 msec の間スタックすることになる。これは様々な処理の性能に支障をきたす。よって、MCPS-DATA.confirm は IEEE 802.15.4 スタックからのメッセージを

MessageQueue で待つという形で実装した。MCPS-DATA.indication コマンドはそもそもデータを受信した際に発行されるコマンドであるため、いつ発行されるか全く予想できない。よって、MessegeQueue を利用した実装が妥当であると考え、そのように実装した。

IEEE 802.15.4 スタックを NES-SOURCE から利用する上で設定が必要な項目は通信チャネル、Personal Area Network ID (PANID)および 16 ビットショートアドレスである。通信チャネルとは IEEE 802.15.4 プロトコルスタックが実際の通信で利用する周波数(PHY 層)のことである。前述の通り、NES-SOURCE は 920 MHz 帯域を使って通信をする。通信チャネルが違うと物理層が違うので互いに通信ができない。IEEE 802.15.4 では、同じ通信チャネルを更に PANID という概念で複数のグループに分けて使用する。通信は同一 PANID 内でのみ実施し、違う PANID の通信機器とは通信できない。IEEE 802.15.4 では、ハードウェア毎に独自の 64 ビットアドレスを持つ。このアドレスを IEEE64 アドレスとよぶ。IEEE 802.15.4 プロトコルスタックは IEEE64 アドレスを使って通信可能である。しかし、そもそも小さいデータをやり取りするためのプロトコルでアドレスに 64 ビットも利用することは無駄が多い。そこで IEEE 802.15.4 では、IEEE64 ビットアドレスの代わりに、16 ビットのアドレスを使っても通信が可能である。このアドレスを 16 ビットショートアドレスという。16 ビットショートアドレスは同一 PANID の中でのみ有効である。IEEE 802.15.4 では、一般的に IEEE64 ビットアドレスは PANID および 16 ビットショートアドレスを付与する時のみに利用し、通常の通信は 16 ビットショートアドレスを使って行う。

IEEE 802.15.4 のフレーム構成(MAC 層)を図 29 に示す。IEEE 802.15.4 のフレームはフレーム内部の構造を表す FrameControl (サイズ：2 byte)と、フレームを識別するために利用する Data Seaquens Number (サイズ：1 byte)と、PANID および送受信アドレスを格納する Address Information (サイズ: 6 byte もしくは 18 byte)と、送信データ本体である Data Payload(サイズ：～118 byte までの任意のサイズ)と、フレームの改ざんやエラーを検知するための Frame Check Sequence (2 byte)からなる。

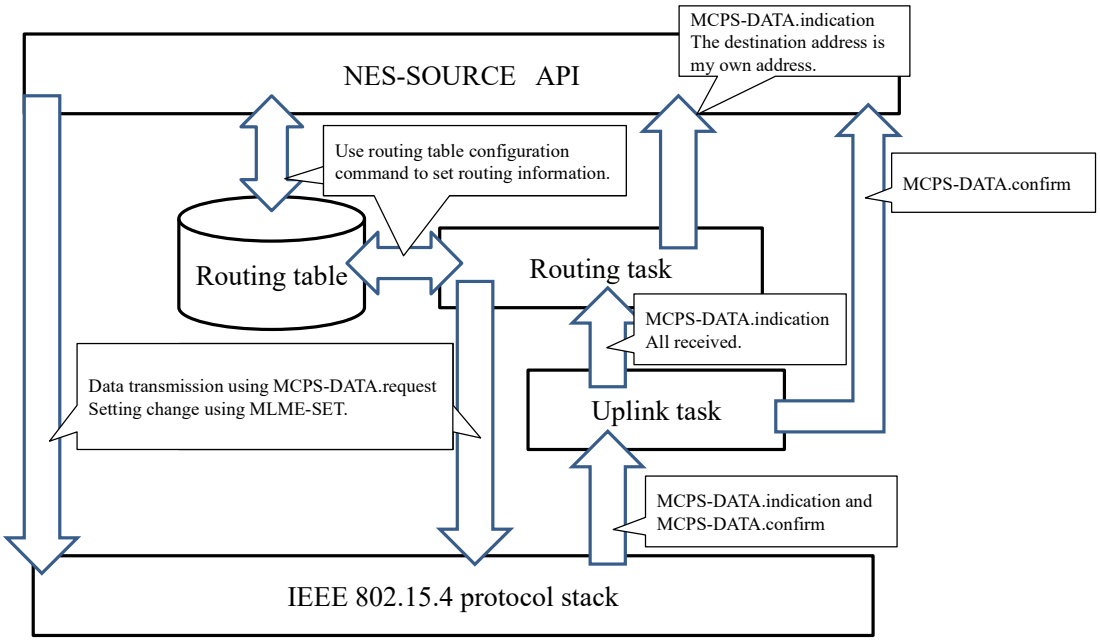

図 30 NES-SOURCE のソフトウェア構成

NES-SOURCE で利用する FrameControl では、MAC 層の ACK の有無および AddressInformation の内容に関する情報が入っている。

IEEE 802.15.4 の Address Information は構成するアドレスサイズの組み合わせにより 4 byte から 20 byte までのサイズを取る。ただし、NES-SOURCE で利用する Address Information 内のアドレスサイズの組み合わせは、共通の PANID 2 byte、送信元 IEEE64 ビットアドレス、送信先 IEEE64 ビットアドレスといった合計で 18 byte になる組み合わせと、共通の PANID 2 byte、送信元 16 ビットショートアドレス 2 byte と、送信先 16 ビットショートアドレス 2 byte の合計 6 byte になる組み合わせの 2 通りである。

4.4.2　ネットワーク層

4.4.2.1 ネットワーク層のソフトウェア構成および動作

ネットワーク層と PHY/MAC 層(IEEE 802.15.4 プロトコルスタック)、ネットワーク層や PHY/MAC 層を使って周辺ノードと通信をする主体であるアプリケーション層といったソフトウェアの構成を図 30 に示す。

NES-SOURCE は他者宛のフレームを MCPS-DATA.request を使って転送する機能、自身宛の受信フレームをアプリケーション層へ通知する機能、受信し

```
while(1){
 Wait_uplink_task_msg(&msg)                    ...(1)
 switch(msg){
 case NESSOURCE_FRAME_TYPE_DATA:          ...(2)
 case NESSOURCE_FRAME_TYPE_DATA_ACK:  ...(3)
    if(msg_is_for_me){                          ...(4)
         send_to_wait _indi_api(msg)          ...(5)
        }else{
         msps_data_req(next_add,msg)          ...(6)
       }
 case SETTING_CHANGE_DATA:                    ...(7)
     if(msg_is_routeinfo){                      …(8)
      cahange_routeinfo(msg)                   ...(9)
     }else{
      mlme_set(msg)                             …(10)
 }
}
```

図 31 ルーティングタスクの擬似コード

た情報から自身の設定(MAC 層の設定やルーティング情報)を変更する機能を持つ「ルーティングタスク」、PHY/MAC 層からの MCPS-DATA.confirm および MCPS-DATA.indication、MLME-SET.(データ受信および設定変更結果通知)をルーティングタスクや上位のアプリケーション層へ振り分ける機能を持つ「アップリンクタスク」、上位から入力されるルーティング情報を記憶する「ルーティングテーブル」、アプリケーション層から NES-SOURCE が提供する機能を利用するための Application Programing Interface(API)である「NES-SOUCE API」からなる。

・ルーティングタスク

ルーティングタスクの動作を示す擬似コードを図 31 に示す。

ルーティングタスクはアップリンクタスクからのメッセージ待ちをしている(図 31(1))。何かメッセージを受信した場合、メッセージの内容がデータパケット(図 31 (2))もしくはデータパケットの ACK (図 31(3))であれば、パケットの送信先をチェックする(図 31(4))。チェックの結果、送信先が自分であった場合は NES-SOURCE API を通してアプリケーション層へ通知(図 31(5))し、そうでなければパケット内のルート情報から送信先アドレスを選んで MCPS-DATA.request を使って次の送信先へパケットを転送(図 31(6))する。メ

```
while(1){
  Wait_MAC_uplink_msg(&msg) ...(1)
  switch(msg){
  case MCPS_DATA_CONF:  ...(2)
    req_handle = get_mcps_data_conf_handle(msg)  …(3)
    send_to_wait_conf_api(req_handle ,msg)      …(4)

  case MCPS_DATA_INDI:  ...(5)
     send_to_routing_task(msg)   ...(6)
  }
}
```

図 32 アップリンクタスクの擬似コード

表 9 ルーティングテーブル操作コマンド

routeinfo_set	Set the routing information in the routing table
routeinfo_get	Get the routing information from the routing table
nextadd_get	Get the next address from the routing information

ッセージの内容が設定変更のパケットであった(図 31(7))場合、パケットがルーティング情報の変更要求パケットであった(図 31(8))場合、内容に沿って自身のルーティングテーブルを変更(図 31(9))する。そうでなければ、PANID 等の設定変更をする(図 31(10))。

・アップリンクタスク

アップリンクタスクの動作を示す擬似コードを図 32 に示す。

アップリンクタスクは MAC 層からのアップリンクメッセージ待ちをしている(図 32(1))。アップリンクタスクは、受信したメッセージ内容が MCPS-DATA.confirm であった(図 32(2))場合、メッセージの中から MCPS-DATA.request 実行時に指定したハンドル値を取り出し(図 32(3))、MCPS-DATA.confirm を待つ NES-SOURCE API へメッセージを通知(図 32(4))する。アップリンクタスクは、受信したメッセージ内容が MCPS-DATA.indication であった(図 32(5))場合、メッセージをそのままルーティングタスクへ通知(図 32(6))する。

```
typedef struct RouteInfo_t{
  uint16_t routeInfo[MAX_ROUTEINFOSIZE];
  uint8_t    routeInfoLen;
  uint16_t distAdd;
 uint8_t  priority;
}RouteInfo_t
```

図 33 RouteInfo 構造体

表 10 NES-SOURCE API

nessource_init	NES-SOURCE Network initialization
nessource_send_req	Data transmission request
nessource_uplink_wait	Wait for messages from NES-SOURCE
nessource_routeinfo_set	Setting routing information
nessource_saddr_set	Setting of 16bit short address
nessource_panid_set	Setting of PANID

・ルーティングテーブル

ルーティングテーブルは、ルーティング情報が集まったものである。ルーティング情報のデータ構造を表す構造体を図 33 に示す。ルーティング情報には、少なくともルートの最終到達地点の 16 ビットショートアドレス(distAdd)、経路情報(routeInfo)、経路の長さ(routeInfoLen)、ルートの優先度(priority)といった情報が含まれる。

ルーティングテーブルやルーティング情報を扱うためのコマンド表 9 に示す。routeinfo_set コマンドは、引数としてルーティング情報を取り、ルーティングテーブルに対してルーティング情報を登録する機能を持つ。routeinfo_get コマンドは、引数として最終到達地点アドレスと、ルートの優先度を持ち、ルーティングテーブルからルーティング情報を取得する機能を持つ。nextadd_get コマンドは、引数としてルーティング情報を持ち、ルーティング情報の経路情報から、自身の次のアドレスを取得する機能を持つ。

・NES-SOURCE API

NES-SOUCE API を表 10 に示す。nessource_init コマンドは NES-SOUCE の初期化コマンドである。具体的には、本コマンド発行時には NES-SOUCE 内のタスクや API 間の通信で利用する MessageQueue やルーティングテーブルを初期化する。

nessource_send_req コマンドはデータを送信するためのコマンドである。nessource_send_req コマンドは引数として、送信データ、送信先アドレス、ハンドル値をとる。nessource_send_req コマンドは、発行されるとまず、送信先アドレスとルート優先度の値を元にルーティングテーブルから routeinfo_get コマンドを使ってルーティング情報を取得する。その後、取得したルーティング情報から nextadd_get コマンドを使って、MAC 層を使ってデータを送信する送信先アドレスを取得する。最後に、取得した送信先アドレスと送信データとハンドル値を引数として、MAC 層に対して MCPS-DATA.request コマンドを発行し、データ送信をする。ACK タイムアウトの時間までに NW ACK packet を受信できない場合、経路を変更してパケットの再送をする。

nessource_uplink_wait は NES-SOUCE からのメッセージ待ちをするためのコマンドである。具体的には、ルーティングタスクから通知される自身宛のデータパケット情報と、アップリンクタスクから通知される nessource_send_req コマンドの発行結果(MCPS-DATA.confirm)を待つ。nessource_send_req コマンドを連続して発行する場合、その送信結果は発行順に上がってくるとは限らない。その場合、nessource_send_req コマンドの発行結果内のハンドル値を調べることで、nessource_send_req コマンドとの紐づけをする。

nessource_routeinfo_set コマンドは、ルーティング情報を引数とする、ルーティングテーブルに対してルーティング情報を追加するためのコマンドである。nessource_routeinfo_set コマンド内部ではルーティングテーブルを操作するための routeinfo_set コマンドを発行する。

nessource_saddr_set コマンドは、16 ビットショートアドレスを引数とする、MAC 層の 16 ビットショートアドレスを設定するためのコマンドである。nessource_sadder_set コマンド内部では、IEEE 802.15.4 のコマンドである MLME-SET.request を発行する。

nessource_panid_set コマンドは、PANID を引数とする、MAC 層の PANID を設定するためのコマンドである。nessource_panid_set コマンド内部では、

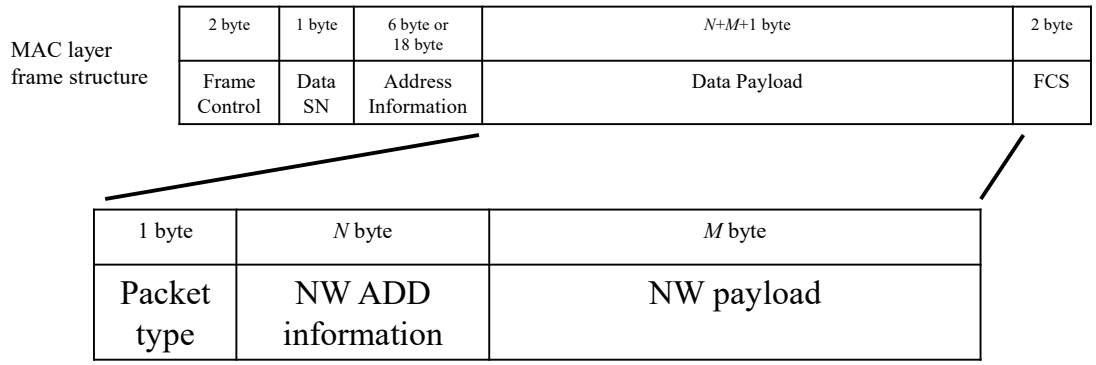

図 34 ネットワークレイヤのパケット構造

表 11 NES-SOURCE パケット種別

Data packet	Data transmittion packet
NW ACK packet	Network layer Ack
Routeinfo set packet	Setting routing information
Short ADD set packet	Setting of 16 bit short address
PAN ID set packet	Setting of PANID

IEEE 802.15.4 のコマンドである MLME-SET.request を発行している。

PANID や 16 ビットショートアドレス、ルーティング情報は、初期値として予め入力しておくことが可能である場合と、ハードウェアの設置後、改めて設定したい場合がある。前者の場合は API を使ってアプリケーションから設定できれば便利であり、後者の場合は無線を使っての設定ができると便利である。こういった要求に対応するため、NES-SOUCE では、PANID、16 ビットショートアドレス、ルーティング情報を、NES-SOURCE API もしくは無線の設定用パケット(後述)のどちらを使っても設定できる。

4.4.2.2 パケット構造

NES-SOURCE が扱うパケットの種類およびパケット構造を表 11 と図 34 に示す。

NES-SOURCE では、データ通信をするための Data packet、Data packet

の ACK である NW ACK packet、ルーティング情報を設定するための Routeinfo set packet、16 ビットショートアドレスを設定するための ShortADD set packet、PANID を設定するための PAN ID set packet という 5 種類のパケットを利用できる(表 11)。NES-SOURCE の packet は、パケットの種類および NW ACK 要求の有無を示す Packet type と、ネットワークレイヤで利用するアドレス情報を示す NW ADD information と、Packet type 毎に内容が違う NW payload からなる(図 34)。

NES-SOURCE のパケット内の Packet type が Data packet もしくは NW ACK packet である場合、NW ADD information にはパケットの送信先までのルーティング情報が入っている。Packet type が Rurteinfo set packet、Short ADD set packet、PANID set packet である場合、NW ADD information には送信元および送信先の IEEE64 ビットアドレスが入っている。Routeinfo set packet、Short ADD set packet、PANID set packet は、Data packet や NW ACK packet と違い、1 HOP 通信のみ対応する。Packet type が Data packet の場合、NW payload には送信データが入る。Packet type が ROuteinfo set packet の場合、NW payload にはルーティング情報が入る。Packet type が Short ADD set packet である場合、NW payload には 16 ビットショートアドレスが、Packet type が PANID set packet である場合、NW payload には PANID が入る。これら設定用のパケットのペイロードは、パケットを受信した NES-SOURCE プロトコルスタックの設定を、ペイロードの値に変更するために利用する。

4.4.2.3 通信経路切り替え機能

NES-SOURCE のネットワーク層は GinMAC と同じく、ルーティング情報を外部から入力する固定経路型のソースルーティング方式である。ソースルーティングとは、ルーティング情報を送信元がパケットに記述しておくルーティング方式である。送信元が通信経路を決定できるので予め遅延を推測できるという利点がある。

また、NES-SOUCE は、ある 1 つの送信先ノードに対して優先度をつけた複数のルートを設定できるという特徴がある。NES-SOURCE は上位レイヤからの送信要求(nessource_send_req コマンドの発行)があった場合、まず一番優先度が高いルートを使ってパケットを送信する。ネットワーク層でパケット送信失敗を検知した場合、NES-SOURCE は次の優先度のルートを使ってパケット

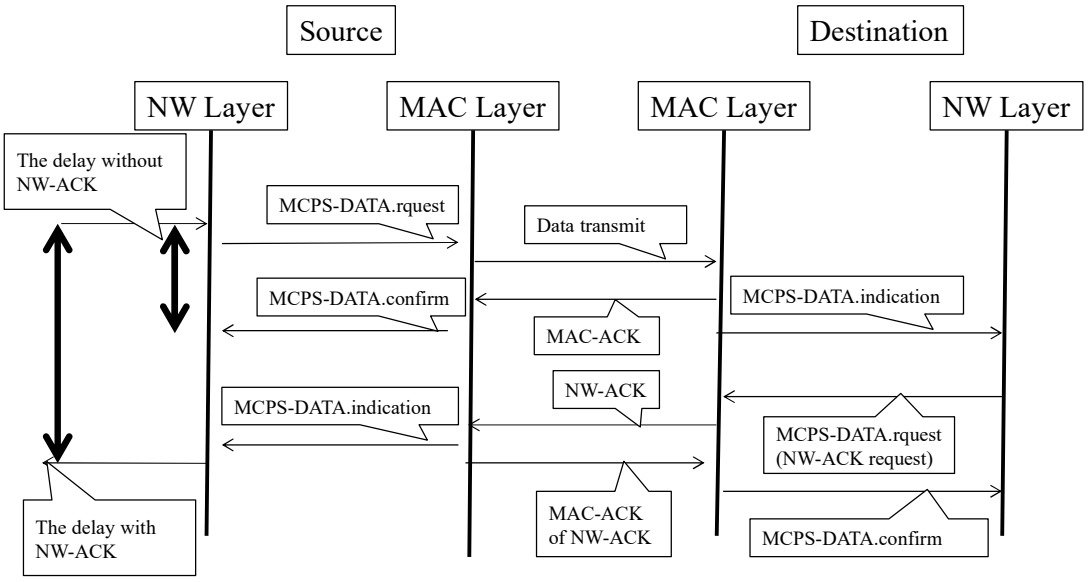

図 35 NW-ACK 利用時の通信シーケンス

を再送する。

NES-SOURCE ではデータの到達確率を上げるため、MAC 層での ACK 通信とネットワークレイヤでの ACK 通信を併用する。ネットワークレイヤでの ACK 通信は、ネットワークレイヤでの通信経路切り替えを実施するために必要である。しかし、ホップ数が 1 HOP である経路でネットワークレイヤの ACK を利用すると、1 HOP の通信到達を確かめるために MAC 層での ACK によるデータ到達確認の後、ネットワークレイヤの ACK 通信を使ってのデータ到達確認という 2 つの通信をすることになる。通信の到達確率向上の観点からすると、ホップ数が 1 HOP である通信経路の通信の到達は MAC 層の ACK 確認で十分である。1 HOP 通信の際はネットワークレイヤの ACK を省略すると、ネットワーク層の ACK 通信が実施されない分だけアプリケーションレイヤから見た通信遅延が削減される(図 35)ので、プロトコルスタックの低遅延化にも貢献できる。

以上のことから、NES-SOURCE では、上位レイヤから指示された送信先への通信ルートが 1 HOP であった場合、ネットワークレイヤの ACK 通信を利用せずに MAC レイヤの ACK 通信呑みを利用してデータの到達を確認するように設計した。この場合、MAC レイヤの ACK 通信に失敗すると直ちに次の優先度のルートに経路を変更して再送を実施する。

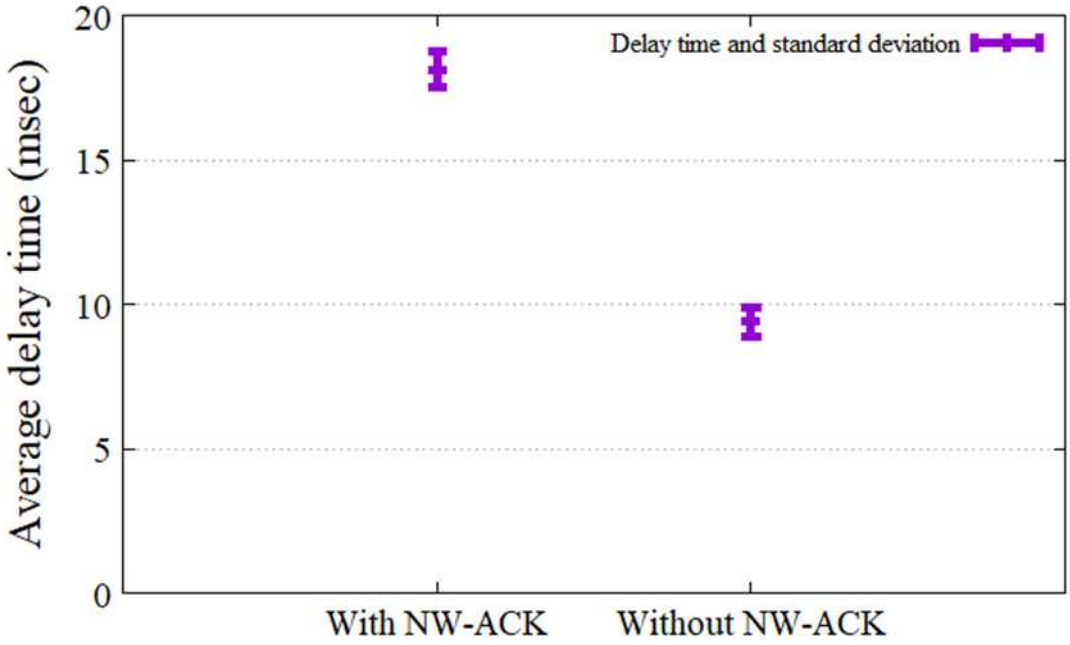

図 36 NW-ACK の有無と平均通信時間の関係

この設計の有効性を示すため、ネットワーク層の ACK の有無と上位層での通信完了確認までにかかる期間の関係を測定した。結果を図 36 に示す。測定時の PHY・MAC 層の条件は表 7 に従い、単位バックオフ時間は 0.2 msec に設定した。また、測定は各 300 回ずつ施行した。NW-ACK 要求有りの場合の 1 HOP 通信時の通信完了確認までにかかる期間は、平均 18.14 msec であり、標準偏差は 0.6 msec であった。一方、NW-ACK 要求無しの場合の通信完了確認までにかかる期間は平均 9.4 msec であり、標準偏差は 0.5 ms であった。1 HOP 通信において NW-ACK を取り除くことで通信時間を約 48%低減できることがわかる。この実験により、1 HOP 通信時のデータ到達確認を MAC 層の ACK のみで行うことは、NES-SOURCE の低遅延化に貢献することが明らかになった。

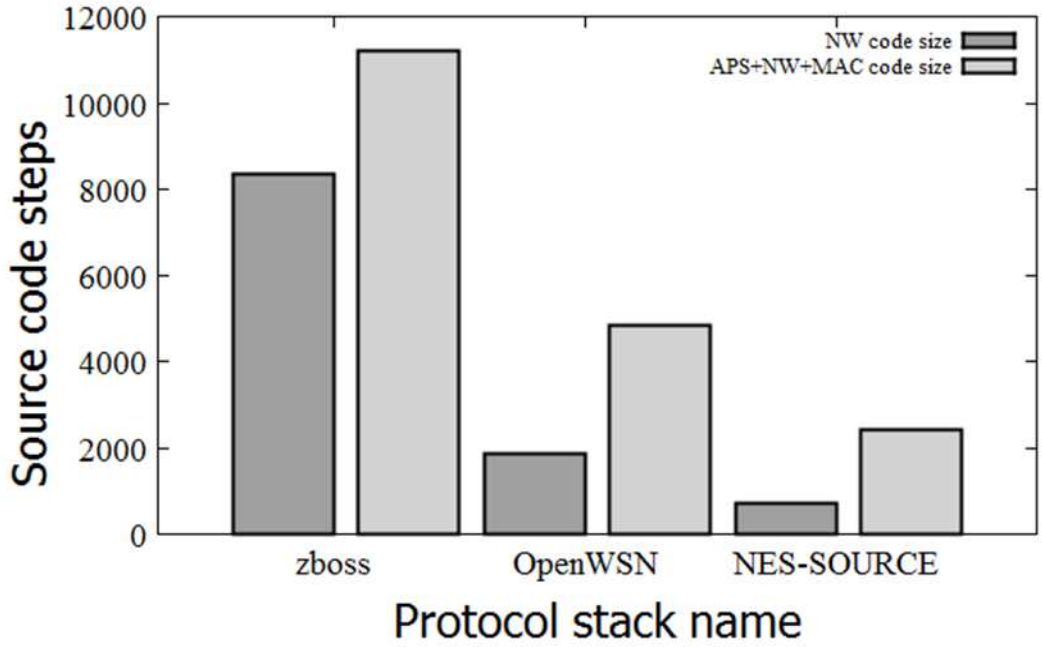

図 37 プロトコルスタックのコードサイズ比較

4.5 実装評価

4.5.1 通信実験環境

提案手法の検証のため、OKI 製 MH920-Node-232 [35]上で NES-SOURCE を実装し評価した。MH920-Node-232 は 32 ビットマイコンである Coretex-M3 を搭載した IEEE 802.15.4g 対応のセンサノードである。ソフトウェアの開発は組み込み OS として FreeRTOS を、開発言語は C を採用した。なお、実験時のノード間距離はすべて 2 m 程度で実施した。この距離は SmarthopMH の通信可能距離（見通しで 1 km）に比べて十分短い。よって実験系に対する反射等の周辺環境の影響は限定的で、実験の再現性は高いと考えられる。

4.5.2 ソースコードの観点からの評価

NES-SOURCE のソースコードをソフトウェア・エンジニアリングの観点から評価した。

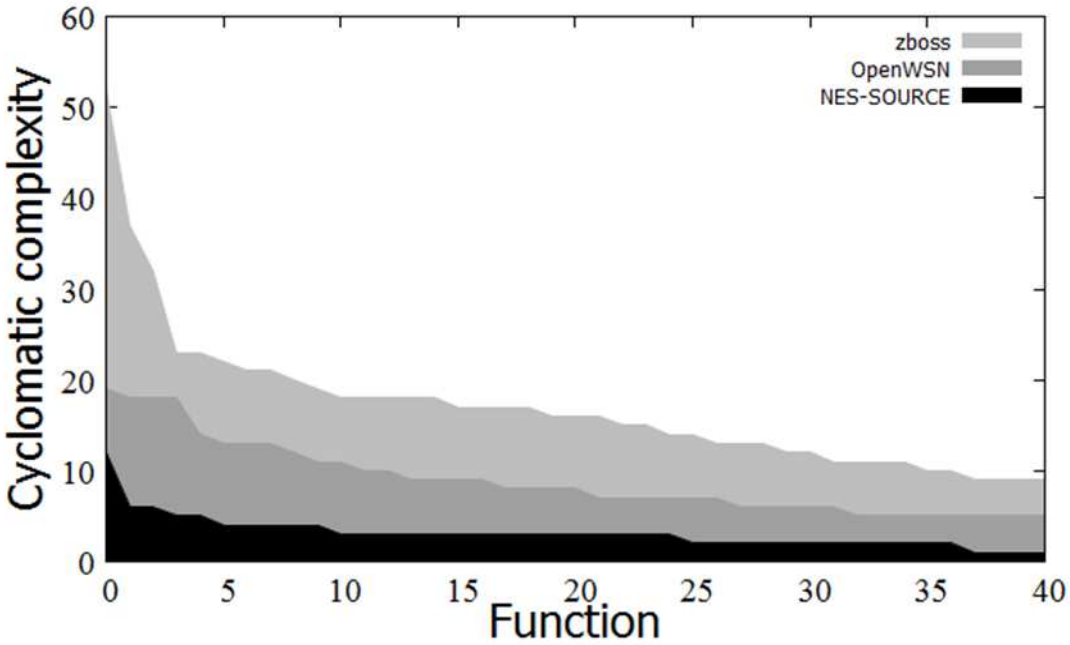

図 38 プログラムコードの複雑さの比較

NES-SOURCE ソースコードは、WSN および WCN のプロトコルスタックと比較しても非常にコンパクトである。ソースコードサイズの比較を図 37 に示す。

一般的なセンサネットワークプロトコルの実装として、zboss [43]を参照した。これは ZigBee PRO のオープンソース実装である。zboss（PHY ドライバを除く）のコードサイズは合計で 11217 ステップである。また、NW レイヤのコードサイズは 8366 ステップである。制御無線プロトコルスタックの実装として、OpenWSN [33]を参照した。これは TSCH のオープンソースである。OpenWSN（PHY ドライバを除く）のコードサイズは 4833 ステップである。また、NW レイヤのコードサイズは 1859 ステップである。NES-SOURCE の場合、合計コードサイズ（PHY ドライバを除く）は 2403 ステップであり、うち、NW レイヤのコードサイズは 727 ステップである。

以上のことからコードステップ数だけで比較した場合、NES-SOURCE は他の実装と比較して非常にコンパクトな実装だと言える。

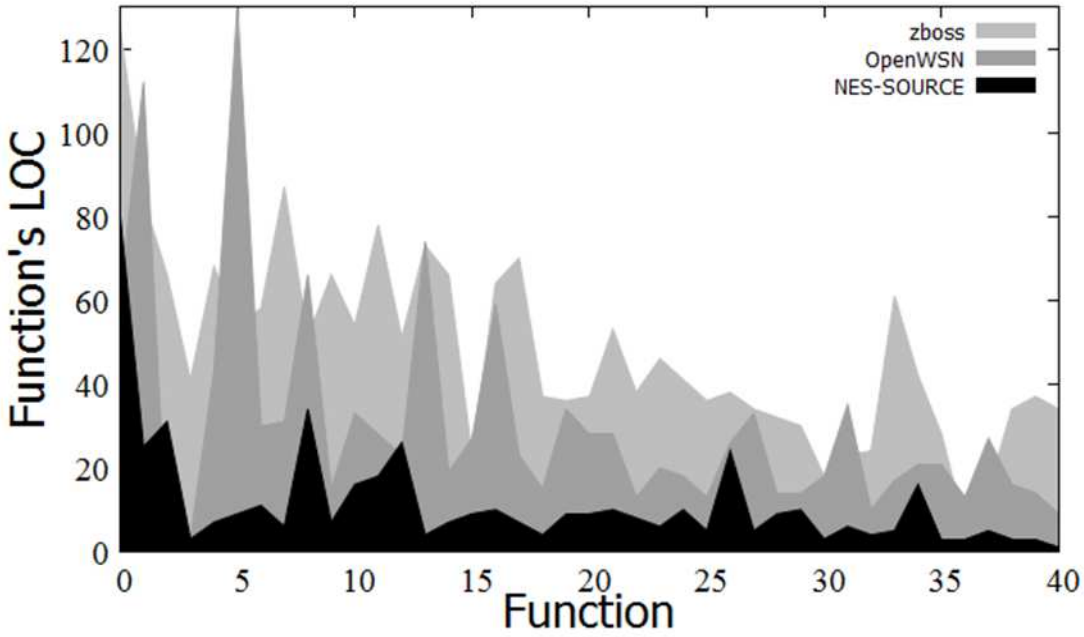

図 39 関数あたりの行数比較

図 38 は、各プロトコルスタックの機能の複雑さを比較した図である。関数の複雑さは、一般的に McCabe 数[44]によって測定することができる。McCabe 数が大きくなるとプログラム処理間の依存関係が複雑になり、プログラムに対して予期せぬ副作用なくコードを追加することが困難になる。McCabe 数が 10 以下の場合、その関数は複雑さの観点から見ると問題はないと言われている。図 38 は、各プロトコルスタックのソースコードにおける、McCabe 数が上位 40 位の関数を比較したものである。この図から、たとえば zboss には 35 個の複雑な関数(McCabe 数が 10 以上の関数)があり、OpenWSN には 10 個の複雑な関数があると言える。一方、NES-SOURCE には複雑な関数は 1 つしか無い。このことからも、NES-SOURCE スタックは、一般的な WSN および WCN プロトコルスタックよりも容易にカスタマイズや調整が可能であると言える。

各プロトコルスタックの関数の内、McCabe 数が大きい関数のステップ数を比較したものを図 39 に示す。一般に、関数あたりのステップ数が少なくなる(目安としては 100 行以下)とコードの可読性が向上し、コードの保守が容易になる。図からわかるように、NES-SOURCE は、他のスタックよりも関数あたりのステップ数が少ない。このことからも、NES-SOURCE は他のプロトコルスタックに比べて保守しやすいと言える。

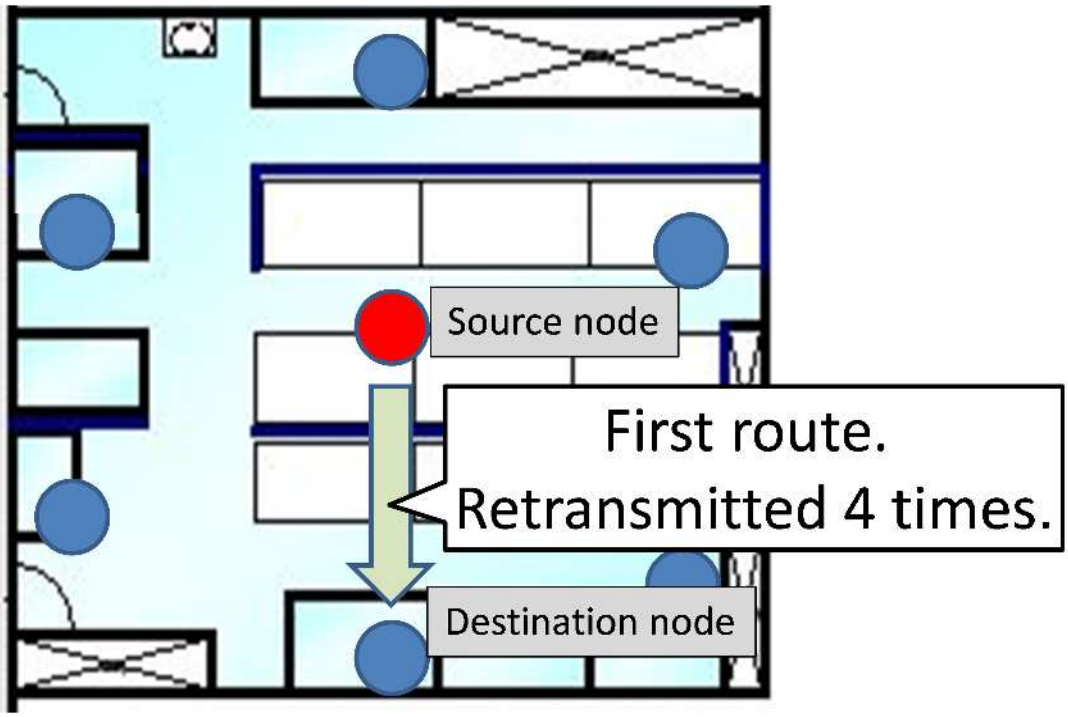

図 40 方式 1：再送 4 回で経路変更なし

4.5.3 測定方法および測定結果

我々は通信遅延と PER を測定するために NES-SOURCE を実装した。実験はトラフィックが無い環境下で通信経路変更機能によって通信遮蔽物を迂回した場合の PER や通信遅延に及ぼす効果を明らかにすることを目的とした。通信経路切り替え機能は、通信環境が変化したときに実現される。そこで、実験は人の出入りによって通信環境が変化する部屋を選んで実施した。

パケットロス発生時の対処方法は以下の 3 パターンに分けて実験を行った。

・方式 1：再送 4 回、経路変更無し：
1 HOP 通信が失敗した時に送信ノードは、1 HOP でフレーム再送する。再送回数の上限は 4 回とする(図 40)。
・方式 2：再送無し、直ちに経路変更：
1 HOP 通信が失敗した時に送信ノードは直ちに 2 HOP の経路を使って再送をする．2 HOP 経路での再送は実施しない(図 41)。
・方式 3：再送 4 回、再送失敗後、経路変更：

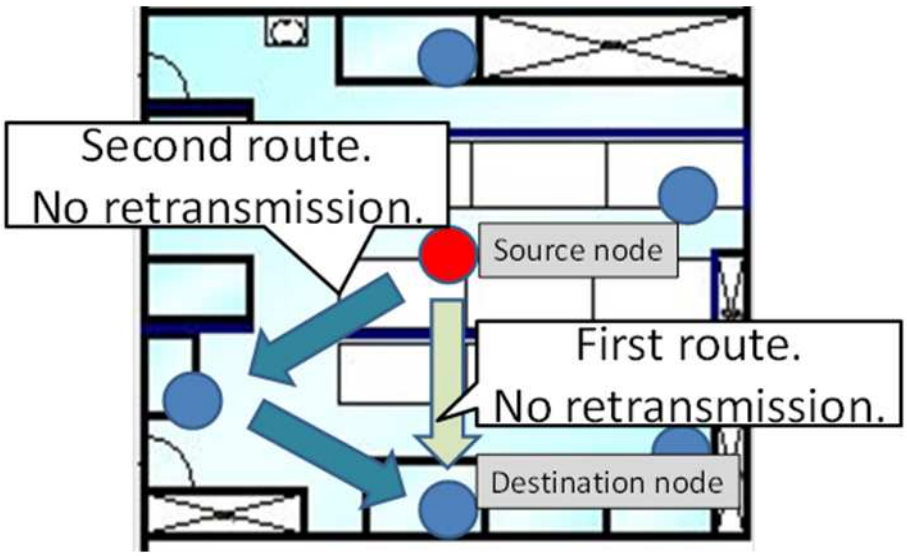

図 41 方式 2：再送なし、直ちに経路変更

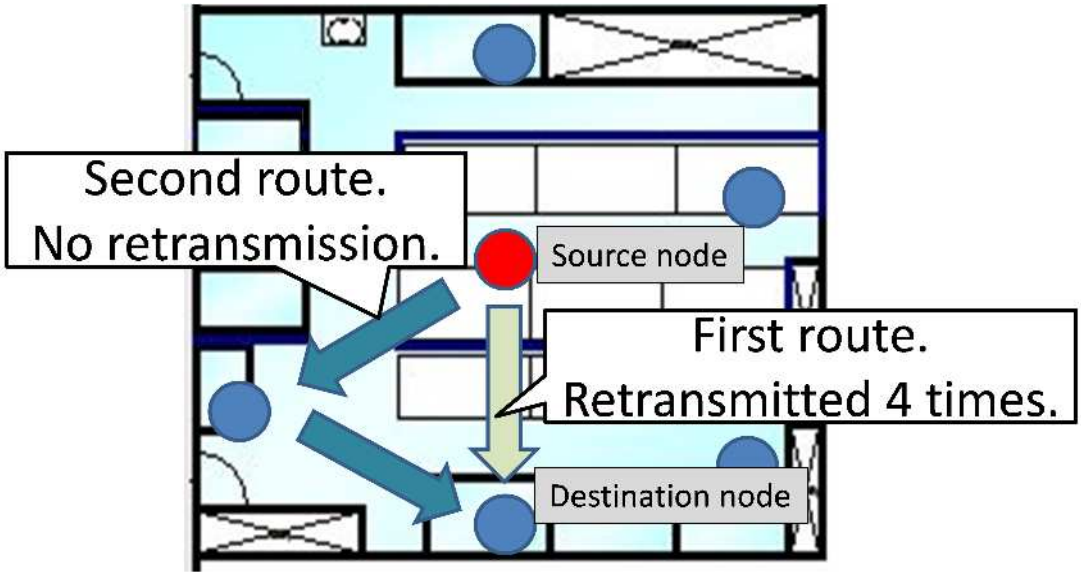

図 42 方式 3:再送 4 回、再送失敗後、経路変更

1 HOP 通信が失敗した際には同一経路で 4 回まで再送し、その後、2 HOP 経路に通信経路を変更する。2 HOP 経路での再送は実施しない(図 42)。

実験は受信側ノードをノード間距離 2 m でヘキサゴナルに配置し、送信ノードを中心に設置して実施した。

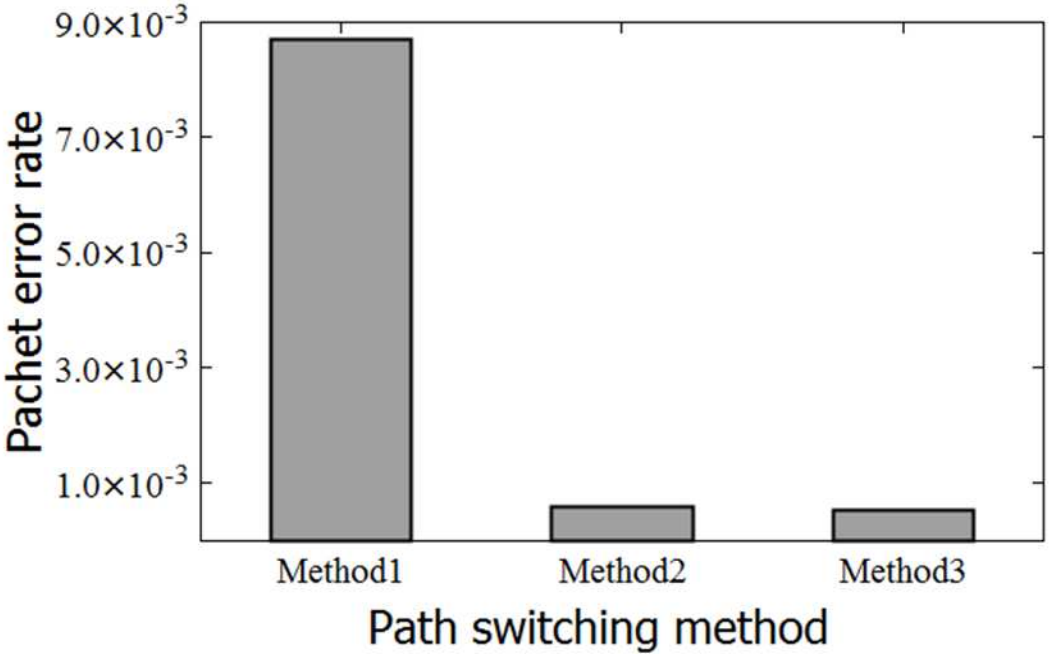

図 43 PER の比較

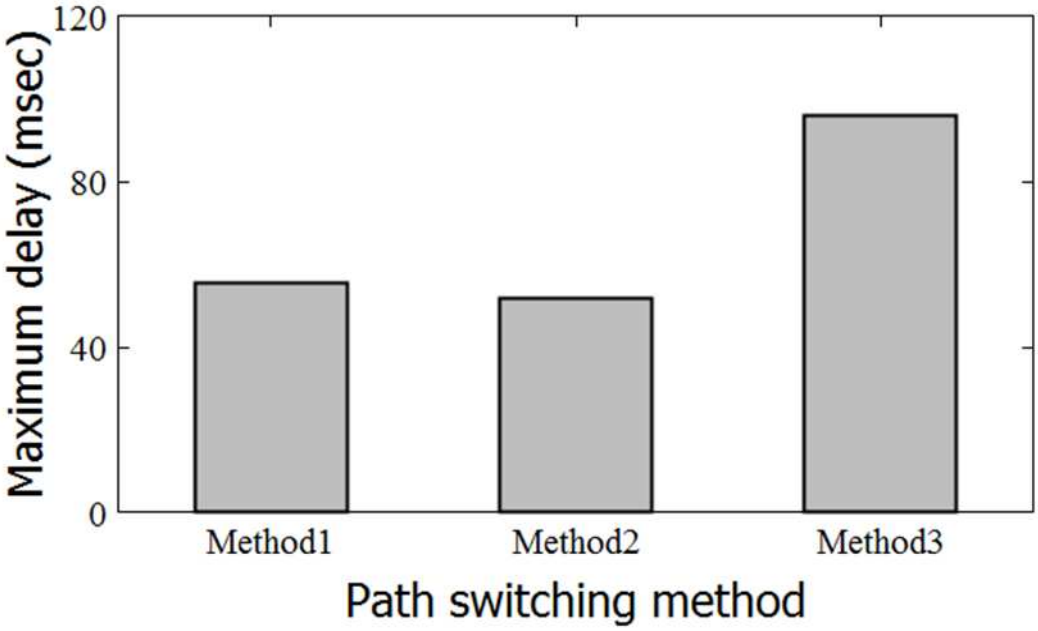

図 44 最大通信遅延の比較

実験時の送信フレームはオーバヘッドを含めて 53 byte、ノード間の距離は平均 2 m、通信系の送信頻度は 1 Hz、室内に入っていた人数は常時 10 人前後、試行回数は各ノード 10000 回ずつ計 60000 回（約 16 時間）である。また、測定は NES-SOURCE の経路変更による遮蔽物回避性能のみを見るために、トラフィ

表 12 PER 計算に利用する値

Number of successful communications method 1 by 1 hop	N_{m1_1hop}
Number of failed communications method 1	N_{m1_failed}
Number of successful communications method 2 by 1 hop	N_{m2_1hop}
Number of successful communications method 2 by 2 hops	N_{m2_2hop}
Number of failed communications method 2	N_{m2_failed}
Number of successful communications method 3 by 1 hop	N_{m3_1hop}
Number of successful communications method 3 by 2 hops	N_{m3_2hop}
Number of successful communications method 3 by 2 hops	N_{m3_failed}
Number of trials	N_{trial}
PER of 1 hop without retransmission under the WCN traffic	$PER_{1hop\_noretry\_c}$
PER of 2 hops without retransmission under the WCN traffic	PER_{2hop_c}
PER of 1 hop with 4 retransmissions under the WCN traffic	$PER_{1hop\_4retry\_nc}$

ックは発生させずに実施した。

実験の結果、PER は 8.71×10^{-3}、 5.83×10^{-4}、 5.33×10^{-4} (図 43)となり、最大遅延時間は 55.37 msec、51.84 msec、 95.93 msec (図 44)となった。また、方式 2 では 1 HOP 経路の利用回数は 58138 回、2 HOP 経路の利用回数は 1827 回であった。方式 3 では 1 HOP 経路の利用回数は 59422 回、2 HOP 経路の利用回数は 546 回であった。

4.5.4 測定結果の考察

4.5.3 項の実験の結果、NES-SOURCE の経路変更機能は有効に機能することが明らかになった。今回の実験環境では、パケットロス時の対策として同一リンク再送(方式 1)に比べて経路変更(方式 2)の方が PER の観点(図 43)から有利であることが分かる。更に、「リンク通信失敗後、すぐに経路変更」(方式 2)と「同一リンク再送失敗後、経路変更」（方式 3）を比較すると、最大遅延の観点(図 44)から比較した場合は方式 2 のほうが明らかに優れているが、PER の観点(図 43)から比較した場合、差は小さい。このことより、今回の実験環境ではパケットロス時のリンク再送の効果が低いことが分かる。以上のことから、人などの

通信遮蔽物が通信路に入る環境では、パケットロス時の対策としてリンク再送よりも経路変更のほうがPERの観点および最大遅延の観点から有効であると考えられる。

パケットロスの原因は通信環境の変化とパケット衝突発生による。4.5.3 項の実験では、パケット衝突の発生については考慮されていなかった。よって、より現実的な PER を計算するため、4.3.2 項のシミュレーションで求めた制御無線ネットワークトラフィック下でのパケット衝突発生確率と実験結果に基づき、それぞれ WCN トラフィックにおける NES-SOURCE の PER を以下に計算した。式内の値の意味は表 12 に示す。

$$N_{trial} = N_{m1\_1hop} + N_{m1\_failed} \tag{20}$$

$$N_{trial} = N_{m2\_1hop} + N_{m2\_2hop} + N_{m2\_failed} \tag{21}$$

$$N_{trial} = N_{m3\_1hop} + N_{m3\_2hop} + N_{m1\_failed} \tag{22}$$

式(20)～式(22)の関係から、パケット衝突が発生しない環境下での各方式のPERは以下の式で表せる。

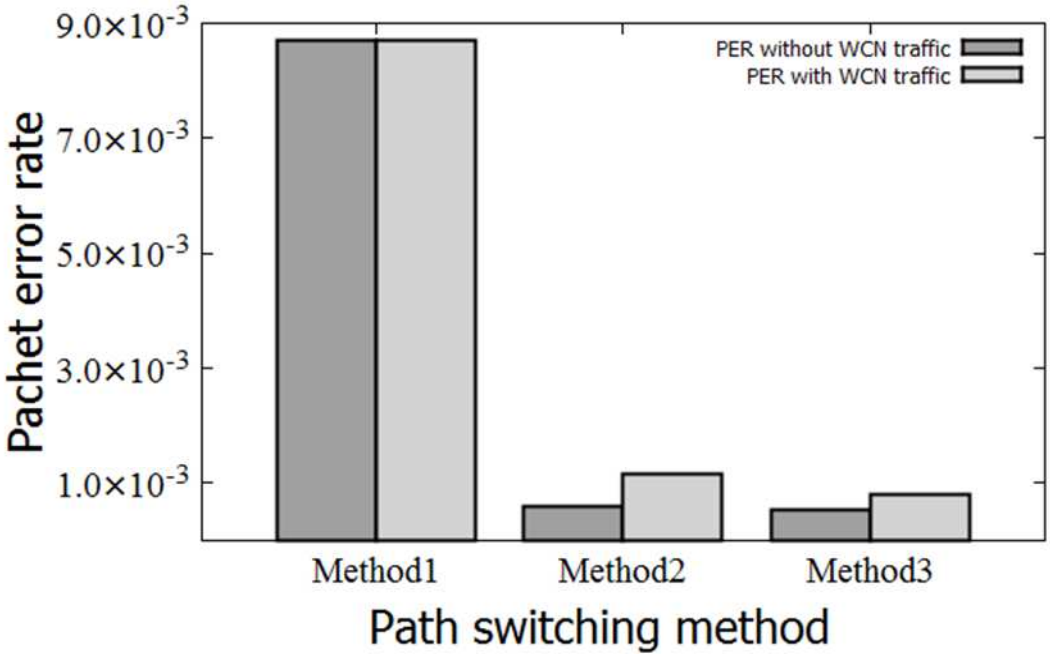

図 45 WCN 通信下での PER

$$PER_{m1\_nc} = N_{m1\_failed} / N_{trial} \tag{23}$$

$$PER_{m2\_nc} = N_{m2\_failed} / N_{trial} \tag{24}$$

$$PER_{m3\_nc} = N_{m3\_failed} / N_{trial} \tag{25}$$

式(23)～式(25)を元にすると、WCN のネットワークトラフィック下での NES-SOUCE の PER は以下の式のように表すことができる。

$$PER_{m1\_nc} = \frac{N_{m1\_failed} + N_{m1\_1hop} \times PER_{1hop\_4retry\_c}}{N_{trial}} \tag{26}$$

$$PER_{m2\_nc} = \frac{N_{m2\_failed} + (N_{m2\_1hop} \times PER_{1hop\_4retry\_c} + N_{m2\_2hop} \times PER_{2hop\_noretry\_c})}{N_{trial}} \tag{27}$$

$$PER_{m3\_nc} = \frac{N_{m3\_failed} + (N_{m3\_1hop} \times PER_{1hop\_4retry\_c} + N_{m3\_2hop} \times PER_{2hop\_noretry\_c})}{N_{trial}} \tag{28}$$

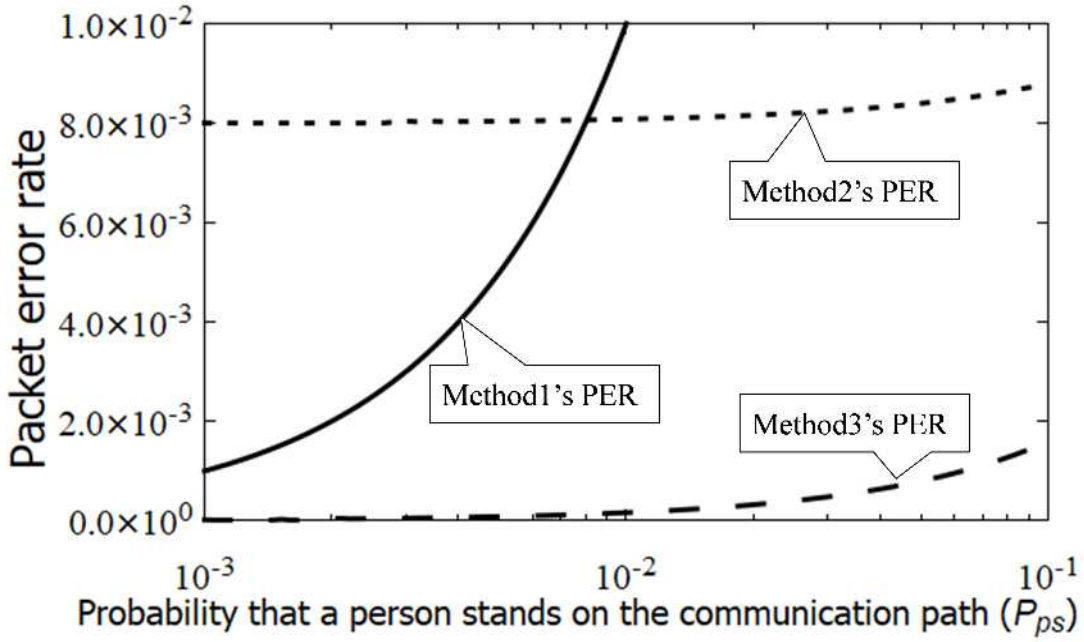

図 46 PER と障害物発生確率との関係

4.3.2 項のシミュレーション結果と、4.5.3 項の実験結果と式(26)～(28)によれば、WCN トラフィック下における NES-SOURCE の PER は方式 1 の場合は 8.71×10^{-3}、方式 2 の場合は 1.16×10^{-3}、方式 3 の場合は 8.05×10^{-4}となった(図 45)。

Zandra らの文献[22]によれば、制御無線ネットワークの PER 要求は10^{-4}のオーダであり、遅延要求は 50～100 ms 程度である。したがって、NES-SOURCE は、対象となる WCN アプリケーションの PER 要件が通常よりわずかに低い場合、利用可能であると言える。実際に無線ネットワークを構築する際は周囲のトラフィックの状況や通信障害物の進入頻度は事前にわからない場合が多い。NES-SOURCE のパケットロス対策はアプリケーション要求や設置環境により柔軟に変更できる。例えば、ネットワークのトラフィックが高いと想定される場合は方式 1 を、通信環境内に人がたびたび立ち入るような環境でネットワークを構築する場合は方式 2 を利用すればよい。最大遅延を犠牲にしても PER を低くしたい場合は方式 3 が適当である。

通信路に遮蔽物が現れた場合に通信が必ず失敗し、遮蔽物によるパケットロスは経路変更により必ず回避できると仮定すると、通信路への遮蔽物出現確率P_{ps}と各方式の PER（$P_{PER\_method1}$，$P_{PER\_method2}$，$P_{PER\_method3}$）の関係を表 12 および以下の式で表すことができる。

$$PER_{Method1} = (1 - P_{ps}) \times PER_{1hop\_4retry\_c} \tag{29}$$

$$PER_{Method2} = (1 - P_{ps}) \times PER_{1hop\_noretry\_c} + P_{ps} \times PER_{2hop\_noretry\_c} \tag{30}$$

$$PER_{Method3} = (1 - P_{ps}) \times PER_{1hop\_4retry\_c} + P_{ps} \times PER_{2hop\_noretry\_c} \tag{31}$$

例えば、$PER_{Method1}$=0.8×10^{-3}、$PER_{Method2}$=1.59×10^{-3}、$PER_{Method3}$＝0（これはシミュレーション結果と同じである）と仮定した場合の PER とP_{ps}の関係を図 46 に示す。この場合、通信路への遮蔽物出現率が 0.8%を超える場合、PER の観点から見ると方式 2 を採用するほうが良い。

第 5 章　結言

5.1　本研究の総括

IoT 技術を利用して様々なセンサから情報を収集し、世の中の現象をモニタリングし社会課題を解決することが現実的になってきた。今後、物理空間から集めてきたセンサデータを AI やビッグデータ処理技術を使ってサイバー空間で処理し、結果を自動的に物理空間へフィードバックして社会課題を解決する「データ主導社会」が訪れると言われている。

データ主導社会では物理空間とサイバー空間のフィードバックループに人間が介在しない。よって、今までのシステムに比べて、処理できるデータの量は多量(数 GB オーダ以上)に、応答時間は非常に短期間(μsec オーダ)になる。このように、今までに比べて量や速度の点からのデータ処理性能が上がるデータ主導社会を実現するためには、現在よりも「多量のデータを収集できる技術」「低遅延でデータ通信する技術」が求められる。こういった技術は論文レベルでは多数検討されてきたが、「標準準拠」「ハードウェアの調達容易性」「低保守・運用コスト」といった産業界からの要求に答えるという観点からの検討はあまり進んでいなかった。

本論文では、データ主導社会を実現するための IoT 無線通信技術である「長期間連続動作して多量のデータを収集するための省電力 MAC プロトコル」および「低遅延のデータ通信が可能なリアルタイム NW プロトコル」を設計、実装し、評価した。設計に際しては産業界で現実的に利用できることを考慮し、前述の 3 つの観点を特に考慮に入れて検討した。

「長期間連続動作して多量のデータを収集するための省電力 MAC プロトコル」に関しては、長期間の連続運用を必要とするインフラ監視システムのための標準技術完全準拠の MAC プロトコルである NES-MAC を提案し、その有効性を検証した。近年、センサの駆動電流は 10～20 μA と小さくなっている。これらの駆動電流は無線通信部の駆動電流である 20 mA に比べて非常に小さい。よって、IoT センサの連続動作を実現するためには無線通信部の省電力化が重要である。

我々はまず、対象となるインフラ監視システムの要件を明確にした。次に、要求仕様から導かれる消費電力と通信リンク成功確率の観点から、標準化された省電力無線通信方式である CSL と RIT について議論した。その結果、データ

通信間隔が 1 時間を超えても CSL 同期通信を維持できれば、要求仕様を満足することがわかった。しかし、一般的なハードウェアで利用される CXO のクロック精度は 30 ppm 程度である。よって、一旦 CSL 同期通信を確立しても通信間隔が 20 分程度空くと CSL 同期通信を維持できない。CSL 同期モードを維持するために、クロック補正機能を備えた NES-MAC を開発した。データ通信間隔が屋内インフラ監視に必要なフレーム送信間隔（約 1 時間）の 3 倍以上であっても、NES-MAC が同期を維持できることを確認した。さらに、CSL 同期モードのウェークアップフレーム送信期間を約 5 倍に延長することにより、NES-MAC を、温度変化が激しい屋外インフラ監視アプリケーションにも使用できることを示した。これらの実験結果から、NES-MAC を用いた場合、2800 mAh (CR123A 型電池 2 本程度)で約 10 年間の連続運転が可能であることが明らかとなった。

NES-MAC は IEEE 802.15.4g および CSL といった標準技術に完全準拠している。また、省電力効果を発揮するために特別なハードウェアや調整が必要ない。よってハードウェアの調達が容易である。さらに、NES-MAC は通常の CSL との通信時にも省電力効果を発揮できるため、システムを増強する際に特別な調整は必要ない。よって、保守・運用コストも低くなる。

以上のことから、NES-MAC は、データ主導社会を実現するために必要な 3 つの要件を満たした、最も現実的で優れた省電力 MAC プロトコルであると結論付けることができる。

「低遅延のデータ通信が可能なリアルタイム NW プロトコル」に関しては、アクセス方式として、扱いやすい CSMA/CA を採用した制御無線向けネットワークプロトコルである NES-SOURCE を開発し、その有効性を検証した。制御無線では、遅延保証が重要な要求事項である。パケット衝突発生によるパケットロスは遅延発生の大きな原因である。アクセス方式として、パケット衝突が発生しない TDMA を利用するとパケット衝突発生による遅延を防ぐことができる。このことから、多くの制御無線プロトコルはアクセス方式として TDMA を利用して実装されてきた。しかし、TDMA 方式は、スロット管理や同期維持等、動作させるためにシステムの観点から制御を必要とする部分が多く、利用が難しい点があった。そこで、アクセス方式として CSMA/CA を利用した、コンパクトな制御無線プロトコルスタックである NES-SOURCE を開発した。CSMA/CA は、方式としてパケット衝突は発生するが、動作させる際にはシステムとしての観点からの制御は不要であるので、TDMA に比べて扱いやすいという利点がある。

我々はまず、一般的な制御無線の利用条件および要求仕様を明らかにした。

その結果、一般的な制御無線の利用条件下にて、アクセス方式として CSMA/CA を利用した場合のパケット衝突発生率は、制御無線の要求仕様に比べて小さいことが明らかになった。このことにより、通信路の環境変化によるパケットロスを回避する方法があれば、CSMA/CA を利用した場合にでも制御無線の要求仕様を満たすことができることがわかった。TDMA を使用する制御無線プロトコルは、通信経路を複数持つことにより、通信経路の環境変化によるパケットロスを減らす。一方、CSMA/CA を利用した場合には、通信路を複数持つとパケット衝突率が多発するので同じアプローチを取ることはできない。

CSMA/CA を利用した制御無線向けネットワークプロトコルである NES-SOURCE は、通信が失敗した場合に通信経路を高速に切り替えることにより、通信経路の環境変化によるパケットロスを減らしている。NES-SOURCE の実装は、アクセス方式として CSMA/CA 方式を採用した結果、コードサイズは約 2700 ステップと TDMA ベースの制御無線プロトコルスタックの半分以下に収まった。さらに、NES-SOURCE は従来の WCN スタックよりもコードの複雑さが低く、カスタマイズして素早く調整することができる。

我々は、通信路の環境が人の出入りにより頻繁に変化する環境で実験を行い、NES-SOURCE の高速な通信経路切り替え機能が効果的に機能することを確認した。例えば最もバランスの取れた方式 2 の場合、実験環境下での PER は 5.83×10^{-4}であり、最悪遅延は 51.8 msec となった。対象としている制御無線の許容通信遅延時間はそれぞれ 100 msec 以下であり、許容 PER は 10.0×10^{-4}程度とされているため、制御無線に適用できるだけの信頼性とリアルタイム性を得ることができたといえる。実験結果によれば、NES-SOURCE の通信障害検出とパス切り替えが高速であり、通信遅延が一般的な制御無線の要求を満足することを示している。

また、人が立ち入ることで通信路の環境が変化する確率、衝突発生による PER、および NES-SOURCE が選択できる経路切り替え方式と、各々PER の関係について明らかにした。このことより、NES-SOURCE を使用するシステムを設計する場合、通信頻度および物理環境におけるパケットロス発生確率を元に、システムの要求仕様を満たすために最適な通信経路切り替え方法を事前に決定することが可能となった。このことより、NES-SOURCE を使って工場のバルブ制御等のアプリケーション構築が可能になったと言える。

従来、制御無線システムを構築する場合には、アクセス方式として TDMA を採用することが一般的であった。これは調達コストや保守・運用コストが高く、利用が難しかった。しかし、我々の実験結果により、制御無線システムのアク

セス方式として調達が容易で保守・運用コストが低い CSMA/CA 方式が利用できると結論付けることができた。

NES-SOURCE は標準技術である IEEE 802.15.4g および IEEE 802.15.4d を使った無線機と相互通信が可能である。また、特別なハードウェアや調整は不要なので、ハードウェアの調達も容易である。さらに、扱いやすい CSMA/CA を利用しており、ソースコードもコンパクトなのでアプリケーション毎の調整もしやすく、保守・運用コストも低くなる。

以上のことから、NES-SOURCE はデータ主導社会を実現するために必要な 3 つの要件を満たした、最も現実的で優れた制御無線向けのネットワークプロトコルであると結論付けることができる。

NES-MAC および NES-SOUCE は通信技術として求められる機能に加え、「標準準拠」「ハードウェアの調達容易性」「低保守・運用コスト」といった、データ主導社会実現のための IoT 無線技術に必須となる条件も満たしている。今後、NES-MAC および NES-SOURCE はデータ主導社会実現のために様々な局面で利用されると考えている。

5.2　今後の課題

NES-SOURCE を更に実用的な無線通信プロトコルとするためには､以下の 2 点について研究開発を進める必要がある。

(1) ルーティング情報の自動設定

NES-SOURCE が対象とする制御無線ネットワークの構成ノード数は最大 25 台程度である。これは一般的なセンサネットワークで想定されているネットワーク規模に比べて小さい。しかし、この規模であってもすべてのノードに対してルート情報を手動で入力するのは煩わしい。したがって、ルーティング設定を自動的に実施するシステムを開発する必要がある。現在、地図データを用いてノードの設置位置からルーティング情報を生成し、ノードへ設定するツールの開発を検討している。

(2) セキュリティ

ネットワークを正しく動作させるためにはルーティング情報をノードに対して正しく書き込む必要がある。特に、ルーティング情報のメッセージ認証は不可欠である。現在、公開鍵暗号方式を使った経路情報書き込み手法を検討中である。

謝辞

本研究に関して、終始温かいご指導とご鞭撻を賜りました直接ご指導していただいた京都工芸繊維大学大学院工芸科学研究科の門勇一教授には深く感謝いたします。先生には技術的なことはもちろん、企業内研究者の考え方や、社会課題を見据えた研究開発テーマの立ち上げ方など、今後、研究者としてだけではなく、企業人として成長するための指標を示していただきました。

慣れない研究を遂行して行くのに際し、私の相談に快く乗ってくださった京都工芸繊維大学の島崎仁司准教授、梅原大祐教授、大柴小枝子教授、上田哲也教授、平田博章准教授に心から感謝いたします。また、論文投稿について色々と相談に乗ってくださった大阪市立大学の原晋介教授および関西大学の四方博之教授にも感謝いたします。

本研究に関して共に考え、共に行動していただいた論文の共同執筆者である、沖電気工業株式会社の松永聡彦様、小林啓洋様、三菱電機株式会社の久保見慎様には特に感謝いたします。また、IEEE 802.15.4 プロトコルスタックの開発、および動作の様子の撮影にご協力いただいた OKI ソフトウェアの久保田剛士様に感謝いたします。

本研究を進めるに当たり温かいご助言をいただきました、沖電気工業株式会社での歴代上司である野崎正典様、久保祐樹様、前野蔵人様、大竹邦之様、中井敏久様、福永茂様、五十嵐譲様、福井潔様、田辺原裕様、西敬様、野中雅人様、山上敬様に感謝します。

研究や業務を進めるにあたり、様々な助言と討論をいただいた沖電気工業株式会社の同僚諸氏に深く感謝します。特に IoT プラットフォーム事業部の山口浩平様は事業部の観点から色々と助言をいただきました。感謝いたします。

関西情報技術士会 PEAK/IT メンバの皆様に感謝します。公益確保および資質向上の責務を長年実践しておられる諸先輩方を間近で見せていただいており、日々大変勉強になっております。

本研究の一部は国立研究開発法人新エネルギー・産業技術総合開発機構(NEDO)の委託業務の結果、得られたものです。委託業務を遂行するための研究コンソーシアム(ライフラインコアモニタリングシステム研究開発(UCoMS))のメンバあるマイクロマシンセンター、産業技術総合研究所、明星電気株式会社、高砂熱学工業株式会社の方々には、特にポンプのモニタリングに必要な通信ネットワークの諸条件に関して様々なご助言をいただきました。深く感謝します。

最後に、本研究の全過程において、常に励まし、支えてくれた家族に感謝します。

参考文献

[1] http://www.ipss.go.jp/pp-zenkoku/j/zenkoku2017/pp29_Report3.pdf (2018/06/16 Access)

[2] 総務省, 情報通信白書平成 28 年版, August, 2017.

[3] http://www.tadviser.ru/images/b/b7/Extreme_automation_and_connectivity_The_global%2C_regional%2C_and_investment_implications_of_the_Fourth_Industrial_Revolution.pdf (2018/06/16 Access)

[4] IEEE Computer Society, "IEEE Standard for Local and metropolitan area networks — Part 15.4: Low-Rate Wireless Personal AreaNetworks (LR-WPANs)", IEEE Std, 2011.

[5] ZigBee Alliance, ZigBee PRO specification, Standard, 2007.

[6] https://aqua-k.jp/goods/goods08.html (2018/06/16 Access)

[7] https://www.arm.com/ (2018/06/16 Access)

[8] https://jp.silabs.com/products/mcu/32-bit/efm32-gecko(2018/06/17 Access)

[9] https://www.raspberrypi.org/ (2018/06/17 Access)

[10] https://www.oki.com/jp/visiot/ (2018/06/16 Access)

[11] https://www.ituaj.jp/wp-content/uploads/2017/03/2017_03-12-ITUclub.pdf (2018/06/16 Access)

[12] http://www.tron.org/ja/ (2018/06/16 Access)

[13] Jamal N. Al-Karaki, Ahmed E. Kamal, "Routing techniques in wireless sensor networks: a survey", IEEE Wireless Communications, vol. 11, no. 6, pp. 6-28, December 2004.

[14] http://ucoms.la.coocan.jp/ (2018/06/16 Access)

[15] Richardo Silva, "GINSENG-performance control in wireless sensor networks", Real-World Wireless Sensor Networks 2010, Lecture Notes in Computer Science, vol. 6511, Springer, Berlin, Heidelberg, pp. 166-169, 2010.

[16] http://www.mlit.go.jp/road/road_e/03key_challenges/1-2-1.pdf (2018/06/16 Access)

[17] http://www.raims.or.jp/ (2018/06/16 Access)

[18] https://www.oki.com/jp/rd/wl/ (2018/06/16 Access)

[19] Ali M. Sadeghioon, Nicole Metje, David N.Chapman, Carl J. Anthony, “SmartPipes: Smart Wireless Sensor Networks for Leak Detection in Water Pipelines”, Journal of Sensor and Actuator Networks, vol. 3, no. 1, pp. 64-78, 2014.

[20] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, W. Pratt, “WirelessHART: Applying wireless technology in real-time industrial process control”, 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 377-386, 2008.

[21] http://www.analog.com/jp/products/rf-microwave/wireless-sensor-networks/smartmesh-wirelesshart.html (2018/06/17 Access)

[22] Pouria Zand, Supriyo Chatterjea, Kallol Das, Paul Havinga, “Wireless industrial monitoring and control networks: The journey so far and the road ahead”, Journal of Sensor and Actuator Networks, vol. 1, no. 2, pp. 123-152, 2012.

[23] Kasun Samarasinghe, Thiemo Voigt, Luca Mottola, Utz Roedig, “Network Coding with Limited Overhearing”, 8th European Conference on Wireless Sensor Networks (EWSN2011), Bonn, Germany, Febrary 2011.

[24] A. Ajith Kumar S., Knut Ovsthus, Lars M. Kristensen, “An industrial perspective on wireless sensor networks — a survey of requirements, protocols, and challenges”, IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1391-1412, 2014.

[25] Petcharat Suriyachai, James Brown, Utz Roedig, “Time-Critical Data Delivery in Wireless Sensor Networks”, Distributed Computing in Sensor Systems, pp. 216-229, June 2010.

[26] IEEE Computer Society, “IEEE standard for local and metropolitan area networks — Part 15.4: Low-rate wireless personal area networks (LR-WPANs) — Amendment 1: MAC sublayer. IEEE Standards”, IEEE Std, 2012.

[27] 藤原純，原田博司，川田拓也，坂元賢太郎，土屋創太，水谷圭一，“IEEE 802.15.4/4e 準拠無線スマートユーティリティネットワーク用超低消費電力 MAC 方式の基礎検討”，電子情報通信学会 技術研究報

告, vol. 115, no. 275, pp. 19-24, October 2015.

[28] 藤原純，原田博司，川田拓也，坂元賢太郎，土屋創太，水谷圭一，“IEEE 802.15.4/4e 準拠超低消費電力 MAC プロトコル F-RIT におけるキャリアセンスの有効性”，電子情報通信学会 技術研究報告，vol. 115, no. 386, pp. 13-18, December 2015.

[29] IEEE Computer Society, “IEEE standard for local and metropolitan area networks — Part 15.4: Low-rate wireless personal area networks (LR-WPANs) — Amendment 3: Physical layer (PHY) specifications for low-data-rate, wireless, smart metering utility networks.”, IEEE Standards, 2012.

[30] A. El-Hoiydi, J-D. Decotignie, “WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks”, 9th International Symposium on Computers and Communications (ISCC 2004), Alexandria, Egypt, July 2004.

[31] Branislav Kusý, Prabal Dutta, Philip Levis, Miklós Maróti, Ákos Lédeczi, David Culler, “Elapsed time on arrival: A simple and versatile primitive for canonical time synchronisation Services”, International Journal of Ad Hoc and Ubiquitous Computing, vol. 1, no. 4, pp. 239-251, 2006.

[32] Lei Tang, Yanjun Sun, Omer Gurewitz, David B. Johnson, “PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks”, IEEE INFOCOM 2011, pp. 1305-1313, Shanghai, China, April 2011.

[33] https://openwsn.atlassian.net/wiki/spaces/OW/overview (2018/06/16 Access)

[34] David Stanislowski, Xavier Vilajosana, Qin Wang, Thomas Watteyne, Kristofer S. J. Piste, “Adaptive Synchronization in IEEE802.15.4e Networks”, IEEE Transactions Industrial Informatics, vol. 10, no. 1, pp. 795-802, February 2014.

[35] https://www.oki.com/jp/920M/mh/unit/ (2018/06/16 Access)

[36] Norman Abramson, “THE ALOHA SYSTEM: Another alternative for computer communications”, ACM Fall Joint Computer

Conference, pp. 281-285, November 1970.

[37] https://www.freertos.org/ (2018/06/16 Access)

[38] http://www.river-ele.co.jp/products/tfx03.html (2018/06/16 Access)

[39] IEEE Computer Society, “IEEE Standard for Local and metropolitan area networks — Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) — Amendment 3: Alternative Physical Layer Extension to support the Japanese 950 MHz bands”, IEEE Std, 2009.

[40] S. C. Ergen, P. Varaiya, “PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks”, IEEE Transactions on Mobile Computing, vol. 5, no. 7, pp. 920-930, July 2006.

[41] E. Felemban, Chang-Gun Lee, Eylem Ekici, “MMSPEED: multipath Multi-SPEED protocol for QoS guarantee of reliability and Timeliness in wireless sensor networks”, IEEE Transactions on Mobile Computing, vol. 5, no. 6, pp. 738-754, June 2006.

[42] M. Strasser, A. Meier, K. Langendoen, P. Blum, “Dwarf: Delay-aware robust forwarding for energy-constrained wireless sensor networks”, International Conference on Distributed Computing in Sensor Systems, Springer, Berlin, Heidelberg, 2007.

[43] A. Cunha, A. Koubaa, R. Severino, M. Alves, “Open-ZB: an open-source implementation of the IEEE 802.15. 4/ZigBee protocol stack on TinyOS”, IEEE International Conference on Mobile Adhoc and Sensor Systems, Pisa, Italy, October 2007.

[44] T. J. McCabe, C. W. Butler, “Design complexity measurement and testing”, Communications of the ACM, vol. 32, no. 12, pp. 1415-1425, December 1989.

[45] 池邉 隆, “IoT の技術動向について”, 特技懇誌, No. 286, pp. 36-46, September 2017.

[46] 独立法人情報処理推進機構, 共通フレーム 2013, March 2013.

[47] http://www.oki.com/jp/920M/sr/ (2018/08/15 Access)

[48] Robert L. Glass, 山浦 恒央, ソフトウェア開発 55の真実と10のウソ, 日経 BP 出版センター, April 2004.

[49] 進士昌明, 移動通信回線の伝搬., 電子情報通信学会, February 1992.

[50] Miklós Maróti, Branislav Kusy, Gyula Simon, Ákos Lédeczi, “The Flooding Time Synchronization Protocol”, The 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, pp. 39-49, MD, USA, November 2004.
[51] Leonard Kleinrock, Fouad Tobagi, “Packet switching in radio channels: Part I — Carrier sense multiple-access modes and their throughput-delay characteristics”, IEEE Transactions on Communications, vol. 23, no. 12, pp. 1400-1416, December 1975.
[52] http://www.tij.co.jp/jp/lit/ds/symlink/tmp102.pdf (2018/08/15 Access)
[53] http://www.analog.com/media/jp/technical-documentation/data-sheets/ADXL372_jp.pdf (2018/08/15 Access)
[54] Giuseppe Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coodination Function”, IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, pp. 535-547, March 2000.

研究業績

・本論文の第 2 章、第 3 章および第 5 章は次の論文からなる。

Yasutaka Kawamoto, Toshihiko Matsunaga, Yuichi Kado.
“MAC protocol with clock synchronization correction for a practical infrastructure monitoring system”,
International Journal of Distributed Sensor Networks, vol. 14, no. 4, 2018.
DOI:10.1177/1550147718773243.

Yasutaka Kawamoto, Toshihiko Matsunaga, Yuichi Kado,
“Clock adjustment for a low power listening wireless infrastructure monitoring system”,
International Conference on Internet of Things and Applications (IOTA 2016), Pune, India, January 2016.
DOI:10.1109/IOTA.2016.7562695

・本論文の第 2 章、第 4 章および第 5 章は次の論文からなる。

Yasutaka Kawamoto, Makoto Kubomi, Yuichi Kado,
“Compact wireless control network protocol with fast path switching”,
Advances in Science, Technology and Engineering Systems Journal, vol. 2, no. 3, pp. 1350-1357, 2017.
Yasutaka Kawamoto, Yuichi Kado.
“NES-SOURCE: Indoor small-scale wireless control network protocol that has a communication failure point avoidance function”
TRON Symposium (TRONSHOW), 2016.
DOI:10.1109/TRONSHOW.2016.7842884

また、本論文は以下の論文を参考論文として作成した。

川本康貴, 松永聡彦, 門勇一,
“クロック補正機能による同期型省電力無線通信方式の性能向上に関する考察”,
電子情報通信学会技術研究報告, vol. 115, no. 189, pp. 65-70, August 2015.
<2015 年度　短距離無線通信研究会　論文賞受賞>

小林啓洋, 久保見慎, 門勇一, **川本康貴**,
“ソースルーティングによる 920MHz 帯無線通信制御システム高信頼化の検

討”, 電子情報通信学会技術研究報告, vol. 114, no. 84, pp. 41-46, June 2014.

久保見 慎, **川本康貴**, 門勇一,
“ソースルーティング機能による 920MHz 帯無線ネットワーク高信頼化の検討”, 電子情報通信学会技術研究報告, vol. 115, no. 275, pp. 25-30, October 2015.
<2015 年度　短距離無線通信研究会　優秀学生論文賞受賞>

Makoto Kubomi, Yuichi Kado, **Yasutaka Kawamoto,**
“Improving reliability of 920 MHz-band wireless networks using advanced source routing function”,
International Conference on Internet of Things and Applications (IOTA 2016), Pune, India, January 2016. DOI:10.1109/IOTA.2016.7562699

Yuichi Kado, Akihiro Kobayashi, Makoto Kubomi, Yuma Nogami, **Yasutaka Kawamoto,**
“Application of economical 920-MHz band wireless communication to power routing in HVDC networks”,
IEEE International Conference on Smart Grid Communications (SmartGridComm 2014), Venice, Italy, November 2014.
DOI:10.1109/SmartGridComm.2014.7007656

川本康貴,
“［招待講演］データ主導社会におけるマルチホップネットワーク”,
電子情報通信学会技術研究報告, vol. 117, no. 178, pp. 35-35, August 2017.