博 士 論 文

# ヘテロジニアスネットワークを模倣した実験環境の構築に関する研究

明石 邦夫

主指導教官　篠田 陽一

北陸先端科学技術大学院大学
情報科学研究科

2017 年 9 月 22 日

# 目次

# 表目次

# 図目次

# 第 1 章

# はじめに

## 1.1　情報と社会

人が持つ根本的な性質として、社会性を持つことが挙げられる。社会とは、人が相互にコミュニケーションをとり、複数人が組織として活動する集団である。人々が社会を形成するためには、情報の伝達が必要不可欠であり、高度に発展した社会は情報社会と呼ばれる。

情報とはそもそも紙や口頭で伝達されていたが、1936 年にアラン・チューリングによって提案されたチューリングマシンにより計算を数学的にモデル化することが可能となった。これにより、我々は情報を紙媒体に記憶しておくだけではなく、コンピュータによって情報を加工することが可能となった。そして、記憶装置の登場によってコンピュータで加工した情報を記憶装置へ保存しておくことにより、我々は情報を紙媒体や口頭で伝達していた現実世界ではなく、情報をデジタル化し仮想世界へ保存し、それを映像として取り出せるようになった。

第 2 次産業革命以降、情報は我々の生活、社会の中で欠かせないものになっている。1991 年、Mark Weiser が予想した「いつでも、どこでも、何でも、誰でも」がネットワークに繋がるユビキタスコンピューティング [1] によって、 2000 年台には様々なデバイスがネットワークに繋がり始めた。このユビキタスコンピューティングの提案によって実現された高度な情報社会を走りに人々はあらゆる場所で情報サービスを利用することができ

るようになった。

高度に発展した情報サービスは、人々の生活基盤の一部として重要な役割を担っている。このような情報サービスを支えているのが、無線ネットワークである。我々が、様々な場所で情報サービスを享受できるのは、持ち運び可能なモバイル端末が無線ネットワークを介してインターネットへ接続されているからこそである。無線ネットワークが普及する以前、情報サービスを享受できる場所は限定的であり、情報サービスの利用も限られていた。しかし、無線ネットワークとそれを利用するモバイル端末の普及によって、場所を問わない情報サービスの利用が可能となった。今や、インターネットへ接続している端末のラストワンマイルは殆どが無線ネットワークを利用していると言っても過言ではない。そして、無線ネットワーク技術は、さらなる利便性の向上のため、新しい方式が次々と新規に提案されている。今後、無線ネットワークはその利用箇所や想定距離、用途に応じて様々な技術が導入され、多種多様な技術が混在するヘテロジニアスなものに変化していくと考えられる。

## 1.2　情報サービスを支える無線ネットワーク技術

情報をサービスとして利活用することで、我々の生活は飛躍的に豊かになった。情報サービスを利用することで、我々は、その場で必要な情報を得ることができる。目的地までの道を交番に聞きに行く必要もなく、他人の連絡先を本で調べる必要もない。これは、情報化によってもたらされた最も大きな特徴であろう。

情報サービスが多様化するに従って、情報を扱うアプリケーションソフトウェアは我々の生活に密着するようになった。このような情報サービスは、社会を支えていると同時に我々の行動にも強く影響を及ぼしている。我々の生活に密着した情報サービスの不具合は、金銭的な損害だけではなく身体へ危険を及ぼす恐れもある。情報サービスの品質は、社会や生活に密着したことで、より高品質なものが求められる。一方で、我々が情報サービスへアクセスする手段は様々である。据え置き型のパーソナルコンピュータは、安定して利用可能な有線ネットワークを利用して情報サービスへアクセスすることが一般的であ

るが、モバイル端末のような持ち運びを想定した小型端末は無線ネットワークを利用したアクセスしか持たない。無線ネットワークは、電波の到達性がある場所であれば利用できるためケーブル配線の繁雑さもなく、利用箇所が幅広いため利便性が高いが、無線ネットワークの通信品質は周辺の建物や人口密度、天候などに左右されるため安定性は低くなる。情報サービスの多様化に伴い、アプリケーションソフトウェアの高い品質が求められる一方で、ネットワーク環境にはより利便性の高いものが使用されている現状がある。

### 1.2.1 情報サービスの複雑化

情報サービスは、現実世界にあった情報をデジタル化することで仮想世界へ移動することから始まった。これまで紙の上で表現されてきた文字や画像、フィルムで投影されてきた映像などはデジタル情報としてコンピュータの中に保存された。情報を現実世界から仮想世界へ移動させたことにより、大量の情報を管理可能となった。情報サービスは、この仮想世界にアクセスすることで、様々な情報を利用したサービスを展開している。情報サービスが広く展開され始めた時代、ネットワークに接続するコンピュータは、パーソナルコンピュータや情報サービスを提供するサーバ、そして携帯電話などであった。そして、提供していたサービスは、Web サービスのような情報の取得やメールサービスや音声通話のような情報の交換が主流であった。つまり、情報サービスはユーザからの要求に対して、情報の提供、交換を主たるサービスとしていた。

これに対し、ユビキタスネットワークでは、電子タグや Wi-Fi などにより人やモノの位置情報を情報として扱うことが可能となった。ユビキタスがもたらした大きな変化は、情報を用いたモノの制御と情報の発信であると言える。モノの制御と情報の発信が可能となったことで、情報サービスが提供するサービスの多様性は大きく向上した。さらに 1997 年に Kevin Ashton が提唱した Internet of Things(IoT) [2, 3] によって、より様々なモノ、人、サービスがネットワークで繋がり、情報を交換することでより大きな価値を生み出すサービスが実現され始めている。

情報サービスは、Web サービスやメール、音声通話が主流であり、End-to-End での情

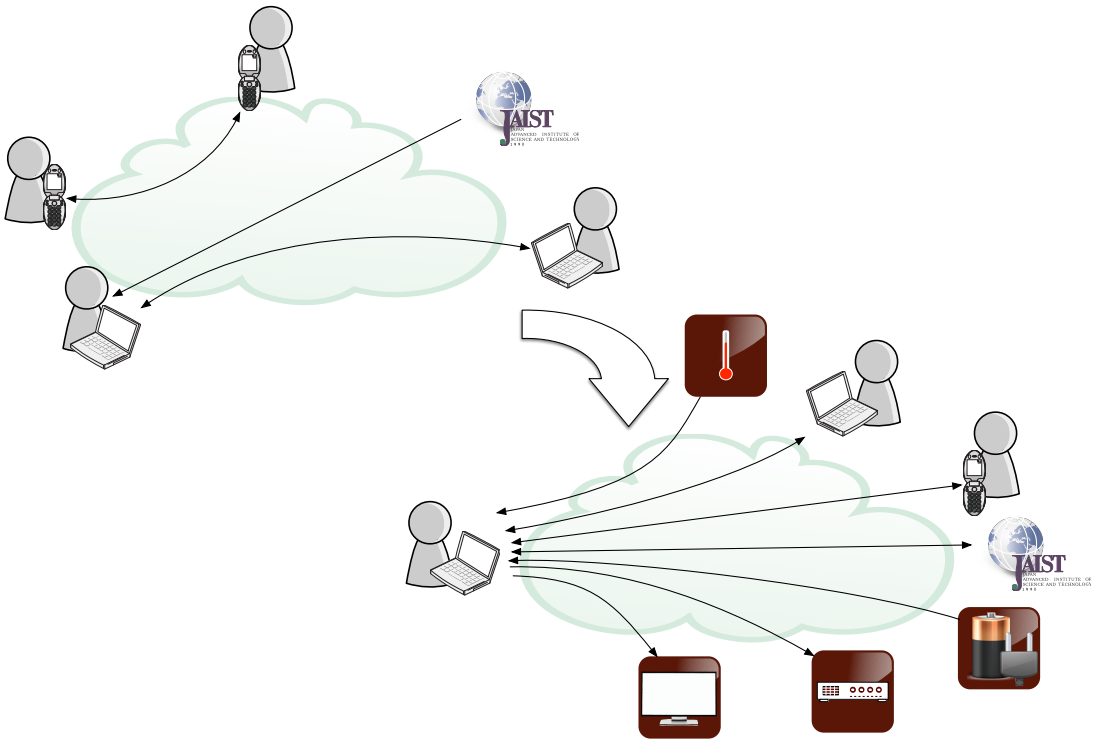


図 1.1: 情報サービスの変化

報交換が主たる利用形態であったこのような端末が提供するサービスは、End-to-End での情報交換ではなく、情報の通知や端末の制御が主な利用形態である。ユビキタスや IoT では図 1.1 のようにエアコンやテレビ、炊飯器のような家電製品、人の動作やモノの状態をセンシングするセンサーデバイスなどもネットワークに接続し、相互に繋がることで情報サービスを提供する。ユビキタスネットワークや IoT の登場によって、情報サービスはコンピュータだけではなく、あらゆるモノ、ヒトがあって成立するものになっている。

### 1.2.2　ネットワーク技術の多様化

ユビキタスネットワークや IoT のような複雑化する情報サービスは、モノやヒトが繋がって成立するが、これらは様々な場所で利用可能なネットワークがあって始めて実現する。インターネット時代と呼ばれた 2000 年代 ADSL や Cable TV のようなブロードバンドサービスが始まり、家庭で広帯域なインターネットが利用可能となった。その一方で、携帯電話が使用するモバイル網や IEEE 802.11 のような無線ネットワーク技術が普

及し、小型端末によるインターネット接続も可能となった。様々なモノやヒトが繋がる情報サービスは、このようなネットワーク技術の革新があったしたからこそ実現したサービスである。

情報社会における様々なサービスを支える無線ネットワークは、我々の生活においても欠かせないものとなっている。このような情報サービスの不具合は、金銭的損害だけではなくは身体的損害も発生しうるため、高い品質、安定性が求められる。そのため、サービス展開前の検証はこれまで以上に重要である。しかしながら、無線ネットワークにはモバイル端末やセンサデバイスのような多種多様な小型端末が多数存在し、各々が相互に繋がる複雑な構成となっており、サービス検証を行うことが難しい。

進化するネットワーク技術は、様々な手法によって安定性、利便性が向上している。その結果、様々な通信プロトコル、通信メディアが展開され、それらが情報サービスを支えるネットワーク環境の中で動作している。特に、ラストワンマイルで利用されている無線ネットワーク技術は、様々な機器、用途と想定した通信技術が多数提案されている。これまでの無線ネットワークでは、インターネット接続に Wi-Fi(IEEE802.11) や 3G、Long Term Evolution(LTE) 新規に が利用され、マウスやキーボードのようなデバイスとコンピュータとの間に Bluetooth が使われている。この使い分けから Wi-Fi は中距離無線、3G や LTE は長距離無線、Bluetooth は近距離無線に分類される。しかし、センサのような小型端末で長距離無線を使用したい場合に、3G や LTE を使用してしまうと、消費電力が高くなってしまう。そこで、低消費電力かつ、近距離無線では満たせない広域をカバーするための通信技術が新規に提案されている。この通信技術には、LoRa [4]、SigFox [5]、WiSun などのように様々な通信規格があり、これらを総称して Low Power Wide Area(LPWA) と呼ぶ。図 1.2 は、現在提案されている無線技術を通信速度と消費電力、通信距離からカバーしている範囲を表した図である。それぞれの技術は、通信距離や用途にとって使い分けることが可能である。つまり、近い将来に実現するであろう情報サービスは、様々な通信モデル、通信技術が混在するネットワーク基盤の上に成り立つであろうと考えられる。

このような多種多様な技術が混在しているヘテロジニアスなネットワーク環境を本論文

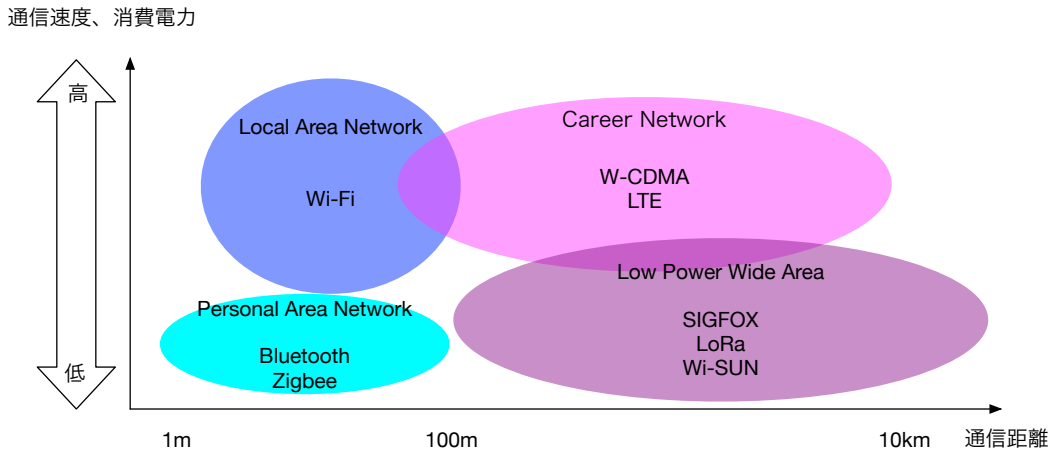



図 1.2: 多様化する無線ネットワーク技術

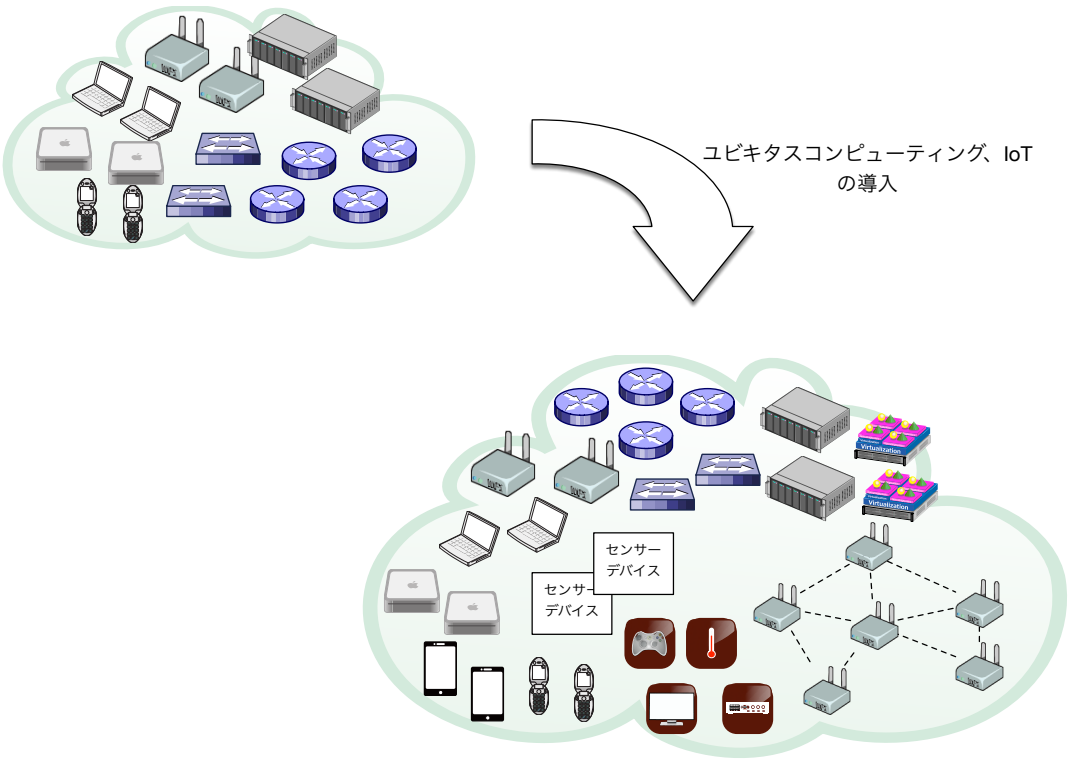



図 1.3: ネットワーク環境の変化

ではヘテロジニアスネットワークと呼ぶ。ネットワーク技術は、新しい技術の提案、導入、淘汰が繰り返し行われ進化している。そのため、社会に導入されるネットワーク技術は、今後も著しく変化し、その数も膨大となっていく予想される。

### 1.2.3 人やモノへの影響

高度に発展した情報社会において、我々は様々な情報サービスを活用して生活している。例えば、我々は目的地へたどり着くまでの経路をカー・ナビゲーション・システムやGoogle Map のようなナビゲーションシステムを活用する。この時、我々はナビゲーションシステムが示した通りの経路を辿り目的地まで移動する。また、エアコンや炊飯器のような家電システムの制御なども情報サービスによって行うことが可能であり、これによって家にいなくてもモノの制御を行うことができる。このように、高度に発展した情報サービスは、我々の行動やモノの状態にも影響を与えている。

また、無線ネットワークを使用するデバイスが増加したことによる影響は既設されているネットワーク機器にも影響を及ぼしている。無線ネットワークでは、周辺の環境による通信品質の変化や、移動による基地局の切り替えを行うハンドオフのような有線ネットワークとは異なる現象が発生する。これらの現象は、有線ネットワークで利用されていた機器に対して大きな影響を与える可能性がある。例えば、Career Grade NAT(CGN) や動画支援のようなミドルボックスは、クライアント、サーバ間に設置されサービスを提供している。セッション管理を行う CGN では、通信品質の変化によるセッション保持時間の増加やハンドオフのような切断、IP アドレスの変更が発生するため、有線ネットワークとは異なる考慮が必要となる。このように、無線ネットワークを用いたデバイスの増加は、我々の生活や行動、そして既設されたネットワーク機器に対する影響が大きい。

## 1.3 情報サービスの検証における課題

これまで述べたように、社会における様々な場面で利用される情報サービスは、人類の生活を豊かにし便利なものとなっている一方で、情報サービスに不具合が発生した際の被害は計り知れないものとなっている。これまでに述べたように、情報サービスは様々なモノや人の行動に強く影響するため、不具合の発生による影響は金銭的損害だけではなく身体障害が発生する可能性もある。そのため、情報サービスは、社会へ展開する前に十分な

検証を行い不具合を排除することが重要である。一般的に検証は、ブラックボックステストやホワイトボックステスト、機能毎の単体テストや結合テストなどがあるが、ネットワーク技術を用いている場合ネットワーク品質も考慮する必要がある。しかしながら、ネットワーク技術の進化に伴い、網羅的な検証が難しくなってきている。この検証の困難性には、ネットワークプロトコル技術の進化と通信メディアの進化の 2 つの側面がある。

### 1.3.1 多種多様な端末が存在する環境

ネットワークプロトコル技術の進化は、既存の通信メディアの上で、信頼性の向上を目的とした様々な技術の提案である。旧来、ネットワーク技術の信頼性は OSI 参照モデルにおけるネットワーク層で行われるものが多数であった。特に、インフラレスな環境でもネットワーク環境を構築するアドホックネットワークは、この傾向が強い。しかし、より信頼性の高いネットワークを実現するため、無線ネットワークに関わる伝搬特性を意識した技術が増加している。ユビキタスネットワークや IoT のような様々なモノが繋がる情報サービスは、People-to-People(P2P) のような複数人の端末間で情報の交換を行うモデルや、ヒトを介在せずに端末が自律的に情報を交換する Machine-to-Machine(M2M) や、People-to-Machine(P2M) のような通信モデルが存在する。

ヘテロジニアスネットワークでは、1 つのネットワーク技術が他のネットワーク技術に影響をあたえることも珍しくない。例えば、IEEE 802.11 は、2.4GHz と 5GHz の電波帯域を使用しているが、IEEE 802.15.1 で定義されている Bluetooth や IEEE 802.15.4 Zigbee も同様に 2.4GHz の周波数帯を使用する。これらの通信技術は、同じ周波数帯を使用していることから相互に干渉しあうため、複数の通信技術を同時に使用すると通信品質が低下する可能性がある。しかし、IEEE 802.11 や Bluetooth は、デバイス間の距離や接続機器、用途によって選択されるものであり、どれか 1 つに統一されるようなものではない。

また、インターネット上では、無線ネットワークのみではなく、有線ネットワークを介して通信を行うことが一般的である。無線ネットワークと有線ネットワークが混在してい

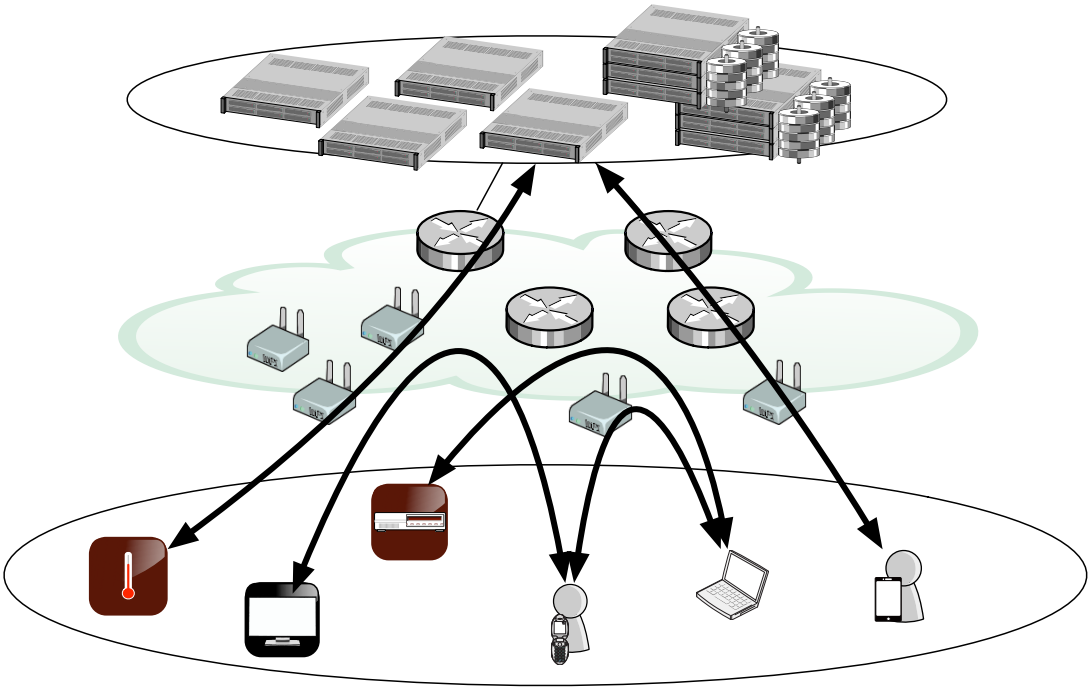


図 1.4: 情報サービスの構成

ることによる影響範囲は、無線ネットワーク内だけではなく有線ネットワーク上で動作している機器にも及ぶ。そのため、単一の無線技術のみに着目した環境では、その影響範囲を網羅的に調査するためには、様々なネットワーク技術を混在した環境での検証が必要である。

### 1.3.2　街や人、モノを含めたサービス検証

これまでの要求から、複雑な情報サービスを検証するためには、モビリティや無線ネットワークではなく、街を構成する建造物や人の動きによって変化する無線ネットワーク、情報サービスと人やモノが相互に影響し合うネットワークテストベッドである必要がある。この様々な要素が影響し合うネットワークテストベッドを実現するために様々な機能を持った単一の構成要素で実現していては、新たな技術を導入する際に大きな改変が必要となってしまう。

### 1.3.3　検証環境に求められる規模

多種多様な情報サービスを支える無線ネットワークにも多様性が求められている。検証対象となるアプリケーションソフトウェアは、検証を行う実験者が持ち込むものであるが、P2P や M2M で利用されるネットワーク環境は検証環境として対応していなければならないだろう。特に、多種多様な無線ネットワーク技術が混在であろう未来のネットワーク構成は、単一の通信技術だけでは不十分である。そして、これらのネットワーク上で利活用される情報サービスは、前節でも述べた通り街規模での展開される。ネットワークテストベッドで再現する無線ネットワークも現実世界と同様の技術を用いて環境再現ができなければならない。

## 1.4　情報サービスの検証基盤

### 1.4.1　ネットワークテストベッドの取り組み

ネットワークテストベッドは、インターネットで利用されるアプリケーションソフトウェアの検証に使われる。アプリケーションソフトウェアを使用する環境を想定し、その環境を再現することで事前に不具合の発見、修正を行うことでリリース前にアプリケーションソフトウェアの品質を向上させることが可能となる。

ネットワークテストベッドが対象とする環境は様々である。データセンタのような有線ネットワークで構築された環境だけではなく、無線ネットワークで構築されるメッシュネットワークや震災時を想定したものまである。ネットワークテストベッドでは、実際に発生したネットワーク環境だけではなく、未来に発生すると予想される環境までも再現する必要がある。

ネットワークテストベッドでは、このような環境を再現するために様々な取り組みが行われている。無線ネットワークのエミュレーション、背景トラフィックの生成、世界中の ISP とその繋がりを再現した箱庭インターネットの構築、対災害を想定した ISP ネットワークと震災後のネットワークなどがある。

## 1.5 ヘテロジニアスネットワークのエミュレーション

これまでのネットワークテストベッドの取り組みでは、特定のシチュエーションを再現している。特に、通信技術は無線ネットワークのみを対象としていることや、有無線混在においても IEEE 802.11 と有線ネットワークの混在環境のみである場合が多い。しかし、無線ネットワークでは Bluetooth や Zigbee など他の通信メディアも提案されており、IoT 機器ではこれらの通信メディアも使用されている。開発者にとって最も注力すべきなのは、検証対象となるアプリケーションソフトウェアであり、ネットワーク環境の構築ではない。検証のためのネットワーク環境および構築に関する手段を提供するのがネットワークテストベッドであり、その存在意義である。

その一方で、ヘテロジニアスネットワークでは、様々な技術が混在しているが、各々の技術が高機能であり再現を行う必要がある。しかし、多数提案されている無線技術を個々に再現してしまうと、混在環境の構築が難しくなってしまう。ヘテロジニアスネットワークを再現するには、特定のプロトコル、ネットワーク技術の制限を受けず、共通のプラットフォームで複数の通信技術を再現しなければならない。

## 1.6 高度化していくネットワーク環境に必要な無線ネットワークエミュレータ

ネットワークテストベッドでは、耐災害を想定した環境や様々な無線技術を使用した環境を再現できるものの特定のネットワーク技術のみ対応しており新しい技術の導入や規模追従性が乏しい。様々なインタフェースが用いられる IoT デバイスは、これまでの無線ネットワークの構成を大きく変化させる。この変化に追従していくためには、無線ネットワークエミュレータが様々なインタフェースを再現する必要がある。また、今後提案されるであろう新しい通信技術に関しても、対応が容易なエミュレーション方法が必要である。そこで、本研究では空間伝搬によるネットワーク特性の再現と有線ネットワーク上で

多種多様な無線技術による通信を可能とする手法を提案する。これまでの無線ネットワークエミュレータは、特定の通信技術を対象としたものが多く、複数の通信技術を用いたヘテロジニアスな無線ネットワーク環境を再現することが不可能であった。これに対し、無線通信で使用される通信メディアに識別子を付与し、イーサネットに含まれていない情報をメタコンテナとして埋め込むことで、有線ネットワーク上で複数の無線技術を用いた通信を可能とする仮想無線ネットワークを実現する。本研究の手法では、今後新しい通信技術が提案されたとしても、メディア識別子とメタコンテナを新たに定義することで有線ネットワーク上での配送方式を変更せずに対応することが可能である。

## 1.7　本論文の構成

本論文では、 2.6 章で無線ネットワークエミュレーション手法とその特徴について述べる。 4.7 章では、本論文で提案する大規模無線伝搬エミュレータである Meteor の設計と実装について述べる。 5.8 章では、仮想無線ネットワークエミュレータである Asteroid の設計と実装について述べる。 6.6 章では、高忠実かつ大規模に展開可能な無線ネットワークエミュレータである NETorium について述べ、 8 章では本論文で着手していない部分、さらに大規模な無線ネットワークエミュレーションを行うために必要となる要件を明らかにする。

# 第 2 章

# 既存研究

無線ネットワーク技術の検証を行うためのシミュレーション手法、エミュレーション手法に関する研究は数多く行われている。本節では、これらの手法について分類を行った上でそれぞれの特徴について述べる。

## 2.1　シミュレーション

ネットワークシミュレーションは、主にモデル検証に用いられる手法である。ネットワークプロトコルやメディア、物理現象などを対象とし、通信状況の計算を行う。

ネットワークシミュレーションにおける計算量は、使用するモデルによって変化する。詳細なモデルを用いれば膨大な計算時間を必要とするが、簡素なモデルを用いれば実時間よりも短い時間で計算結果を得ることができる。また、多くのネットワークシミュレータは、対象とするソフトウェアプログラムに改変を必要とする。ネットワークシミュレータは、無線ネットワークの性能、品質に関わる電波伝搬のシミュレーションを行うことができる。特に電波の伝播経路を追跡または推定し、電波伝搬特性を求めるレイトレーシングは、外部要因となる建造物、地面などによる反射・回析・散乱・透過などを含めたネットワーク品質を扱うことが可能である。

### 2.1.1　レイトレーシング

レイトレーシングには、レイラウンチング法と、イメージング法の 2 種類が広く知られている。レイラウンチング法は、送信点から一定の角度毎にレイを発信し、その軌跡を追跡し受信点に到達するレイを発見する。イメージング法は、送信点、受信点、そして建造物などの反射面の組み合わせから幾何光学的に反射面を求める。

レイラウンチング法は、図 2.1 に示すように、送信点から様々な方向にレイ (光線) を放射し、受信点までの伝播経路を求める。レイの放射方法は、一定角度ごとに放射する方法や、送信点の指向性を考慮した放射方式でもよい。放射されたレイが障害物のような電波を反射させる物体に到達すると、その物体との交点を反射点とし、レイの方向を変更する。レイラウンチング法では、これにより各レイの追跡を行い、受信点までの伝播経路を求める。しかし、放射するレイ数、送信点と受信点の距離によっては受信点に到達するレイを発見できない可能性がある。そこで、受信点の周囲に受信エリアを設け、この受信エリアに到達したレイを電波伝搬経路として用いる。

レイラウンチング法の計算量は、基本的に放射レイの数に依存する。そして、電波伝搬経路の精度は、放射レイの数と受信エリアの大きさによって決定する。放射レイの数が少ない場合、計算量は抑えることが可能であるが、受信エリアに到達するレイを発見できない可能性がある。しかし、受信エリアのサイズを大きく設定すると電波伝搬経路の精度が低下する。送信点から放射されたレイが受信エリアに到達可能であるかは、送受信点間の距離に大きく依存するため、レイラウンチング法を用いる際には、シミュレーション環境に応じて放射レイの数と受信エリアの大きさを適切に設定する必要がある。

イメージング法は、電波伝搬経路の探索に全ての障害物の面に対して到達判定を行い、送信点から受信点に到達するレイを探索する。図 2.2 にイメージング法を用いた際の電波伝搬経路の探索方法を示す。イメージング法では、障害物に対して、鏡像点と呼ばれるイメージを仮定し、受信点までのレイをトレースする。図 2.2b では、障害物での反射回数が 1 回のみの場合のレイをトレースしている。まず、送信点から障害物 A を中心とし

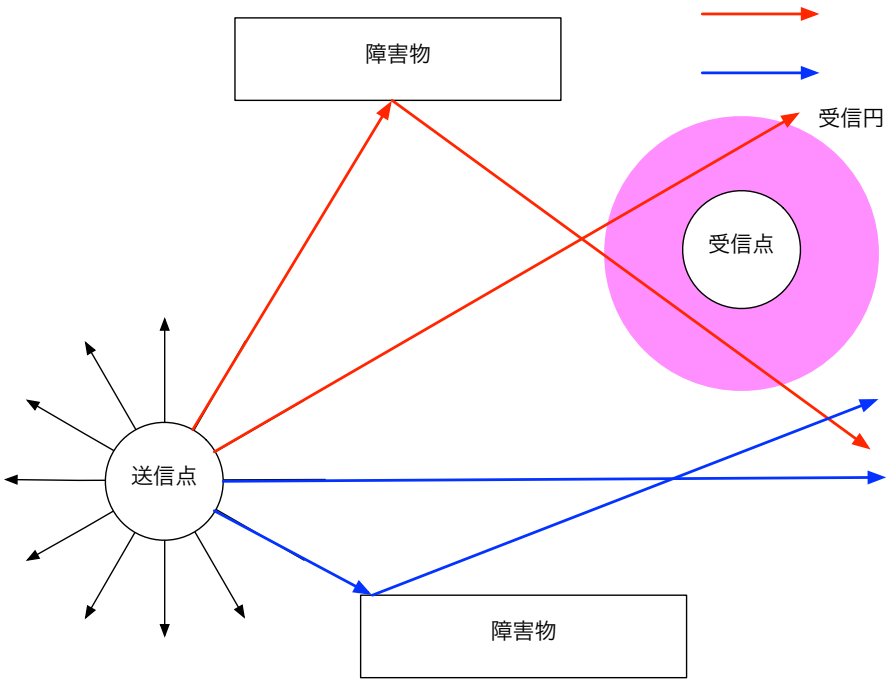
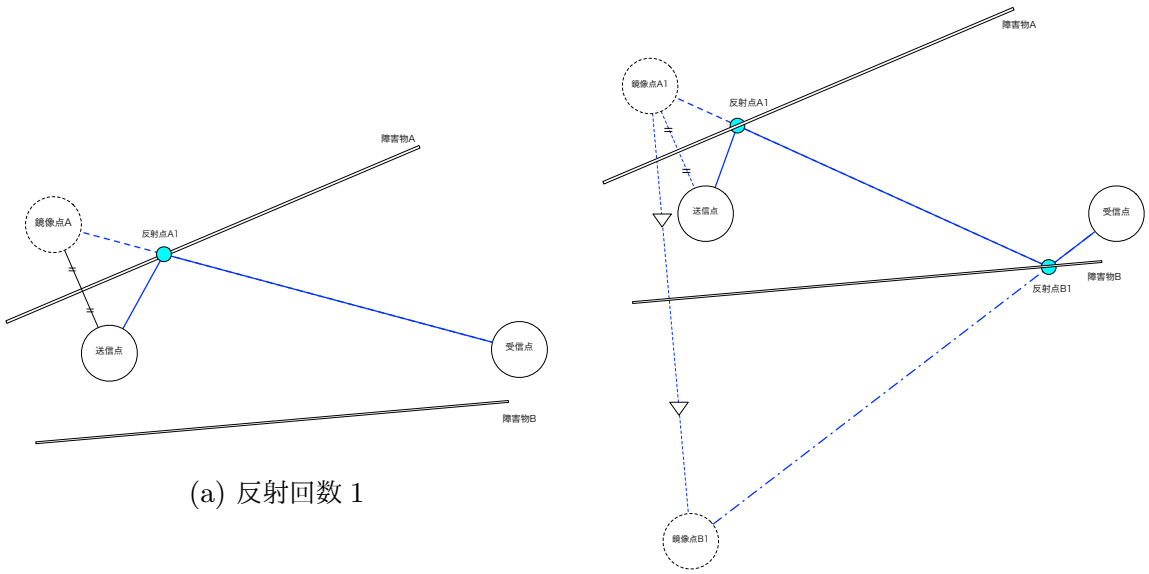



図 2.1: レイラウンチング法による電波伝搬経路の追跡

た対面に鏡像点 A を仮定する。そして、鏡像点 A から受信点までの直線を求める。この直線と障害物 A との間を反射点とし、送信点、反射点、受信点を結んだレイを反射回数 1 回のレイとする。反射回数を 2 回とした場合、鏡像点 A までは反射回数 1 回と同様に求めるが、その後、鏡像点 A から障害物 B を中心とした対面に鏡像点 B を置く。そして、鏡像点 B から受信点までの直線を求め、障害物 B との交点を反射点 B とする。その後、反射点 B から鏡像点 A までの直線を求め、その間に見つかる障害物 A との交点を反射点 A とする。最後に、送信点、反射点 A、反射点 B、受信点を結んだレイが反射回数 2 回のレイとなる。このように、イメージング法では、送信点から受信点までの伝播経路を正確に求めることが可能であるため、シミュレーションの精度が高い。しかしながら、障害物の数を M、考慮する反射回数を N とした場合、$M(M-1)^N - 1$ の組み合わせを経路探索する必要がある。そのため、複雑な地形や障害物が多数存在する環境では計算量が膨大になる。

レイトレーシングによる電波伝搬経路の探索は、アプリケーションソフトウェアの検証ではなく、電波伝搬の際の減衰量やシャドーイング、フェーシングなどの影響を明らかに




(a) 反射回数 1

(b) 反射回数 2

図 2.2: イメージング法による電波伝搬経路の追跡

するために行われる。そのため、検証手法としては送受信間のパスロス計算や屋内外伝播解析などに利用されている。

### 2.1.2　ネットワークシミュレーション

ネットワークシミュレーションは、ネットワークデバイスやプロトコル、アプリケーションをモデル化し、検証を行う手法である。前節で述べたレイトレーシングとは異なり、プロトコルやアプリケーションの評価を行うことが可能である。ネットワークシミュレーションを行うツールには、ハードウェアなものやソフトウェアなものが存在するが、本節ではソフトウェアによって実現されているソフトウェアシミュレータを主軸に述べる。

ソフトウェアシミュレータには、Network Simulator 2(NS-2) [6]、QualNet [7]、OPNET [8] などがある。ソフトウェアシミュレータは、シミュレーションクロックで動作するため、リアルタイム性を持たない。また、ノード数の増加やより複雑なプロトコル処理を行った場合、シミュレーションに必要な計算量が飛躍的に増加する。そのため、ソフトウェアシミュレータを使った研究でも最大で 100 ノード程度 [9, 10, 11] のシミュレー

ション結果となっている。以上のことから、ソフトウェアシミュレータはモデル検査やアルゴリズムのテストに向いている手法である。

アルゴリズムや動作モデルの検証を行うためのソフトウェアとして、Network Simulator version 2(NS-2) [6] や QualNet [7], OPNET [8] などがある。ソフトウェアシミュレータであるため、動作はリアルタイムではなく、シミュレーションクロックである。そのため、ノード数が増加するとシミュレーションに必要な計算時間は飛躍的に増加し、大規模な環境でのシミュレーションを行うと現実的な時間で終わらなくなってしまう。実際にこれらの研究においても 50 から 100 ノードのシミュレーションにとどまっている [9, 10, 11]。また、実際にフレームを送受信して動作するわけではないので、ルーティングアルゴリズムやプロトコルモデルの理論検証に向いている。

ネットワークシミュレータが実世界に近い環境を再現するためには、物理現象やプロトコルの振る舞いを忠実にモデル化している必要がある。しかし、詳細なモデル化は計算量の増加につながるため、現実的な時間でシミュレーション結果を得ることが難しくなる。事実、多くの論文では 100 ノード程度の実験となっている。

一方で、簡素なモデルを用いた場合、シミュレーション結果の信頼性に疑問が残る。

## 2.2 ハードウェアエミュレーション

ハードウェアエミュレーションは、汎用 OS 上で無線インタフェースや小型デバイス向け OS を動作させることで、API の動作までを含めた再現を行う方法である。ネットワークシミュレーションやレイトレーシングとは異なり、PHY エミュレーションや小型デバイスで使用されるマイクロプロセッサのエミュレーションを行うため、アプリケーションコードも実際に展開するものをそのまま使用できる。

### 2.2.1 インタフェースエミュレーション

ハードウェアエミュレーションの方法として、汎用 OS 上で無線インタフェースを再現する方法がある。本論文では、これをインタフェースエミュレーションと呼ぶ。

インタフェースエミュレーションは、無線インタフェースが持つ PHY をソフトウェアで再現し無線フレームを送受信する。インタフェースエミュレーションを行っている実装には、 MAC80211_HWSIM [12] や、BlueZ btvirt [13]、fakelb [14] などがある。MAC80211_HWSIM は、IEEE 802.11 radio を、BlueZ btvirt は Bluetooth を、fakelb は IEEE 802.15.4 の再現を行う。これらのエミュレータで再現された無線インタフェースは、実際のハードウェアと同じ動作をするため、ユーザ空間で動作するアプリケーションソフトウェアの検証に使用できる。

インタフェースエミュレーションは、図 2.3 に示すように、PHY と仮想インタフェースを再現する設計となる。インタフェースエミュレータの処理は、PHY の再現と無線インタフェースのドライバまでとなっており、ドライバを制御は各無線技術の汎用ライブラリが使用できる。そのため、アプリケーションソフトウェアは、実無線インタフェース、インタフェースエミュレータによって生成された仮想無線インタフェースに関わらず動作することが可能である。インタフェースエミュレータを用いた場合、TCP/IP とは異なるプロトコルを用いた検証が可能となるため、無線アクセスポイントや、認証、Personal Area Network(PAN) のような IEEE で規格化されている技術を用いた検証が可能となる。しかしながら、これらのインタフェースエミュレータは、PHY 間でのフレーム送受信と無線インタフェースの生成を主な機能としており、無線空間で発生する電波衝突や、それによる転送レートの制御などは行われない。生成された無線インタフェースを用いた通信は、無線インタフェース間のリンク速度に関係なく無線フレームの伝送が行われ、最大転送速度を超えた値が計測されてしまう。また、PHY 間では無線フレームによる通信が行われるため、複数の無線インタフェース間で通信を行うためには、1 台のノード上でコンテナや仮想マシンなどの仮想化技術を用いる必要がある。このような特性からインタフェースエミュレーションは、小規模かつ自由空間を想定した環境での検証に向いている手法である。

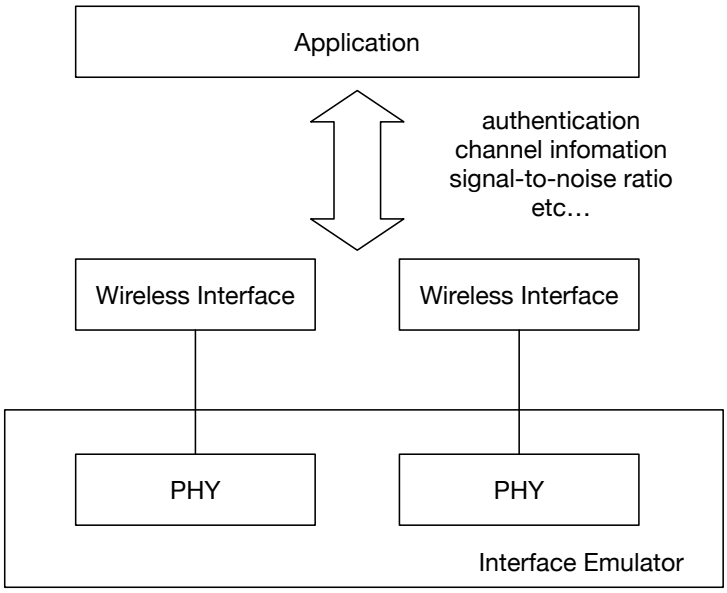



図 2.3: インタフェースエミュレーション

### 2.2.2 デバイスエミュレーション

デバイスエミュレーションは、OS 上に仮想マシンとして小型デバイス向け OS を動作させる手法である。インタフェースエミュレーションとは異なり、汎用 OS が使用されないデバイスのアプリケーションソフトウェアの検証を行うことが出来る。

Cooja [15] は、IoT 向けのオペレーティングシステムである Contiki OS の開発環境に含まれているネットワークエミュレータである。Cooja は、MSPSim と呼ばれるマイクロプロセッサ用のエミュレータを用いて無線ノードの再現を行う。

これらのデバイスエミュレータでは、IEEE 802.15.4 上で使われる 6lowpan のような無線プロトコルが利用可能であり、仮想マシン上で動作するアプリケーションソフトウェアは、無線インタフェースや仮想マシン上で動作している OS の Application Programable Interface(API) が利用可能なため、RSSI や SSID のような情報も利用可能である。Cooja も インタフェースエミュレーション と同様、無線フレームの送受信や端末の挙動を再現し、実デバイスと同じ API を提供することから忠実度が高いと言える。しかしながら、複数のノード間での通信は不可能であるため、デバイスエミュレーション

も、小規模かつ自由空間を想定した環境での検証に向いている手法である。

## 2.3　ネットワークエミュレーション

### 2.3.1　無線ネットワークエミュレーション施設

現実性を重視した無線ネットワークエミュレーションを行う研究に、施設内に専用の無線ノードを設置し、制御することでアプリケーションソフトウェアの検証環境を構築するものがある。

#### ORBIT

Open-Access Research Testbed for Next-Generation Wireless Networks(ORBIT) [16] は、2003 年 9 月に NSF Network Research TestBED(NRT) の下で開始された大規模無線ネットワークテストベッドである。ORBIT は、アメカ Rutgers 大学の Wireless Information Network Labratory に 400(20 × 20) 台の無線端末 を室内に配備されており、その中で無線ネットワーク技術やアプリケーションソフトウェアの検証が可能である。

ORBIT は、屋内に設置した無線端末を制御することで無線グリッドを構築しており、各無線端末は、IEEE 802.11g/a/n/ac や、Zigbee、Bluetooth が使用可能である。また、Software Defined Radio (SDR) platforms (USRP, WARP, RTL-SDR, USRP N210, USRP X310) や、GNU Radio も使用することが可能である。

ORBIT は、無線端末を室内に配備していることから、各端末が移動することはせず、ソフトウェアでアンテナの送信出力を制御することで、ノード間の距離を再現している。また、実際に無線端末を用いるため、高い忠実性を担保しており、展開するプログラムコードも検証用に改変を加える必要がないといったメリットがある。しかし、実際に無線伝送を行うため、実際の無線通信を行った検証ができるが、大規模な検証を行うためには広大な土地が必要となる。また、モビリティや環境情報を再現する場合でも、ORBIT で提供されている環境パラメータ、ノード数しか扱えないため、多様性、規模追従性に欠

ける。

NITOS

NITOS [17] は、Thessaly 大学の Network Implementation Testbed Laboratory が研究開発を行っている実世界と同様の設定でアプリケーションソフトウェアとネットワークプロトコルの検証を可能とする無線ネットワークテストベッドである。有線、無線ネットワークを混在させたネットワーク環境の構築に重点を置いており、様々な環境の再現が可能である。

NITOS は、室内、屋外、オフィステストベッド、クラウドテストベッドの 4 つのネットワーク環境から構成され、室内、屋外、オフィステストベッドでは、Wi-Fi や WiMAX、LTE の無線インターフェイスがサポートされている。屋内環境は、テッサリア大学キャンパスビル内に 50 台の無線ノードで構成され、屋外環境は、50 台の無線ノードが設置されている。また、クラウド環境として複数のサーバが利用可能であり、OpenFlow [18] を用いて Software Defined Network (SDN) の検証も可能としている。NITOS で提供されている無線ノードは、 cOntrol Management Framework(OMF) と呼ばれるオープンソースソフトウェアで管理されている。このように NITOS では、多種多様なノード、ネットワーク技術を用いたアプリケーションソフトウェアの検証が可能なテストベッドであるが、規模追従性の点から見ると大規模であるとは言い難い。

### 2.3.2 伝搬エミュレーション

伝搬エミュレーションは、無線通信における電波伝搬を計算し、その結果からノード間での遅延時間や帯域幅、パケットロス率などの伝搬特性を実際の有線ネットワーク上で送受信するパケットに適用することで、無線ネットワークでの振る舞いを再現するエミュレーション手法である。伝搬特性の計算は、IEEE 802.11 や 802.15.4 などの通信メディアによって計算方法が異なるため、様々な方法が選択されている。

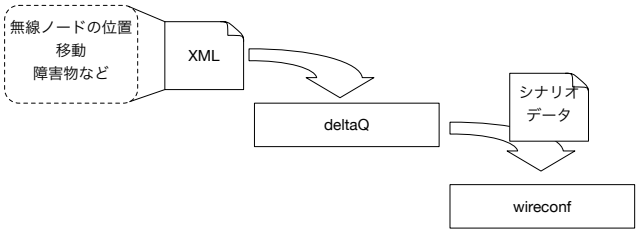



図 2.4: QOMET の伝搬エミュレーションまでの処理

### QOMET

QOMET は、 Beuran らが提案した Wi-Fi や Zigbee のような無線ネットワークにおける伝搬特性を擬似的に再現する伝搬エミュレータである。QOMET には、無線ノードの移動や環境情報から伝搬特性をシミュレーションする deltaQ と deltaQ のシミュレーション結果から有線ネットワーク上で擬似的な無線ネットワークを再現する wireconf から構成される。

QOMET が伝搬エミュレーションを行うまでの処理を図 2.4 に示す。

deltaQ は、XML(Extensible Markup Language) 形式で記述されたシナリオをもとに、無線ノード間の通信品質を計算する。XML には、無線ノードの初期位置や移動情報、周辺に配置されている障害物、電波伝播に関するパラメータなどを記述する。ノードの位置情報は、XY 座標、または緯度経度で表現され、移動情報は QualNet の外部シミュレータの計算結果を用いることができる。環境情報には、deltaQ が扱うフィールドの大きさ、建造物の情報や減衰パラメータ $\alpha$、分散パラメータである $\sigma$ を記述する。deltaQ は、これらの情報から Log-distance Path Loss Model を用いて無線ノードの間の信号減衰を計算する。そして、計算した信号減衰値から通信品質を示す Frame Error Rate(FER) を算出する。以上の処理によって算出したシミュレーション結果を wireconf 用のシナリオファイルとして出力する。このシナリオファイルには、無線ノード間の遅延時間や帯域幅、パケットロス率などが時間毎に記述されている。QOMET では、シミュレーションを行う deltaQ とエミュレーションを行う wireconf に分離しているため、新たな無線メディアが提案されたとしても、deltaQ の改変を行うのみでエミュレーションが行える設

計となっている。

wireconf は deltaQ が出力したシナリオファイルから無線ネットワークの振る舞いを有線ネットワーク上に擬似的に再現する伝搬エミュレータである。wireconf は deltaQ に記述されている遅延時間や帯域幅、パケットロス率を deltaQ の出力ファイルから読み込み、それらを送信元 ID と送信先 ID で示されるノードペア間で送受信するパケットに適用する。

wireconf は、図 2.5 で示すようにネットワークスタック上のネットワーク層で動作する。wireconf は、ネットワーク上を流れるパケットから deltaQ で管理している ID を判断することはできない。そのため、ネットワーク層の識別子である IPv4 アドレスと deltaQ の ノード ID を紐付けるテーブルを持つ。deltaQ が管理しているノード ID と各ノードの IPv4 アドレスを対応付けることで、ネットワーク上を流れるパケットの送信元と送信先のノードを判断することが可能となる。しかし、各ノードが送信するパケットには必ずしも一意な IPv4 アドレスがついているわけではない。ノード間の通信がユニキャストであれば、送信元アドレスと送信先アドレスは一意なものである。しかし、ブロードキャストによる通信が行われた場合、送信先アドレスはブロードキャストアドレスが使用されるため、送信元 ID と送信先 ID のペアで判断することができない。この問題に対して wireconf では、ユニキャストとブロードキャストを 図 2.6 のように扱っている。ユニキャストパケットに対しては、送信元 IPv4 アドレスおよび送信先 IPv4 アドレスのペアからノード間の伝搬特性を反映する。ブロードキャストパケットに対しては、受信したパケットの送信元アドレスを用いてテーブルから送信元 ID を参照する。そして、送信元 ID と 自身の ID を合わせてノード間のペアとし伝搬特性の反映を行う。

QOMET にリアルタイム性をもたせた実装として dynamiQ がある。dynamiQ は、エミュレーション実行時のノードの位置情報を外部シミュレータから入力することで、その時間におけるノード間の伝搬パラメータを計算し、パケットに対して適用する。ノードの移動情報を自身で持たず、リアルタイムにシミュレーションを行うという点以外では、QOMET と dynamiQ は同じ伝搬エミュレータである。しかし、deltaQ の計算時間はノード数によって変化するため、リアルタイム性を持たすためには 100 ノード程度が

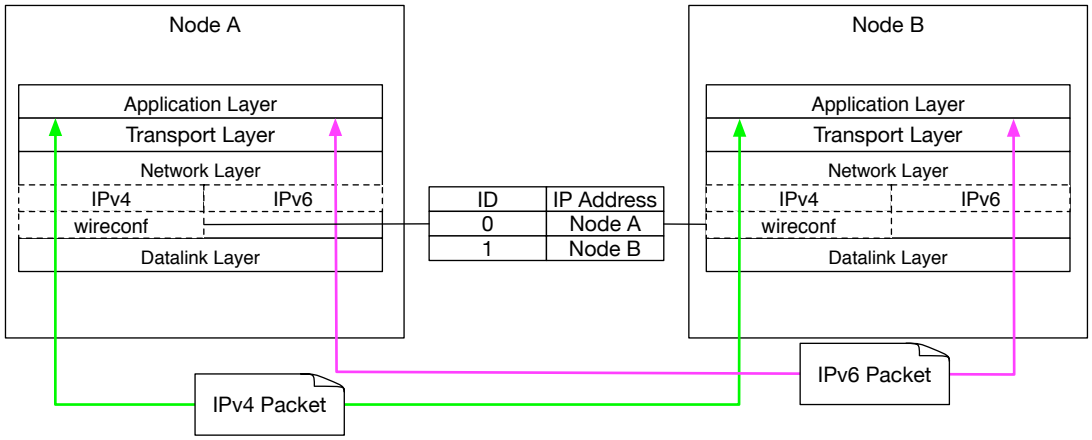



図 2.5: OSI 参照モデルにおける wireconf の動作場所

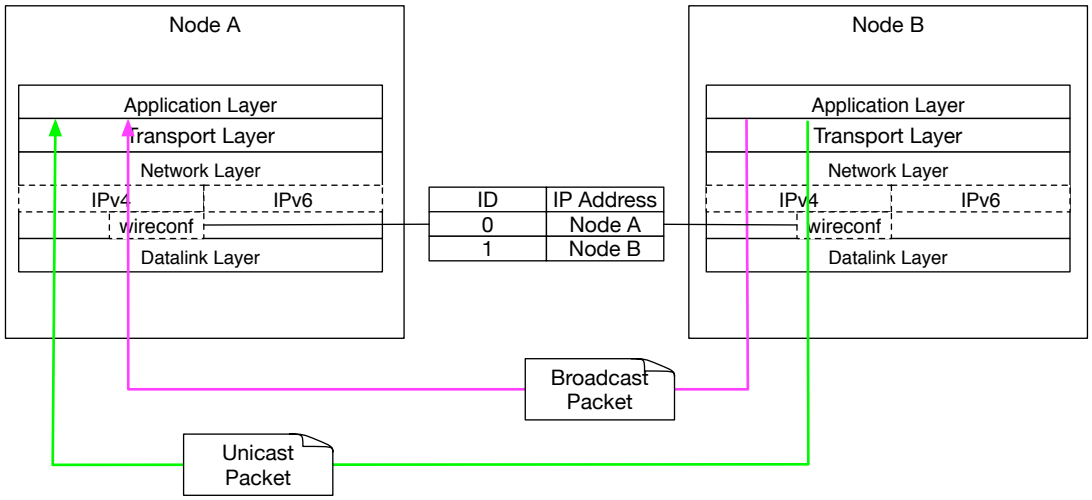



図 2.6: wireconf のユニキャストとブロードキャストの扱い

dynamiQ の最大規模となる。

### MobiNet

MobiNet は、伝搬エミュレータである Modelnet [19] をベースに無線アドホックネットワークに対応させた実装である。そのため、MobiNet のネットワーク構成は、Modelnet と同じものとなる。

Modelnet は、エッジノードとコアノードと呼ばれる汎用ノードとイーサネットスイッチから構成されるクラスタ環境で動作する伝搬エミュレータである。エッジノードは、検

証を行うアプリケーションを実行するノードを指し、エッジノード上では複数の仮想ノード (Virtual Node:VN) が動作する。コアノードは、エッジノード間の無線空間を再現する。エッジノード間の通信は、必ずコアノードを経由する。MobiNet が再現する伝搬パラメータは、遅延、帯域幅、パケットロスそして、MAC レイヤである。

図 2.7 は、MobiNet による伝搬エミュレーションの処理を示した図である。VN から送信されたトラフィックは必ず Core Module を経由する。Core Module では、まず受信したパケットを Packet Filter に通し Routing Module へパケットを転送する。Routing Module は、受信したパケットから送信先の VN までのパスを探索し伝搬エミュレーションを行う Pipe Module へ転送する。Pipe Module では、伝搬エミュレーションを行う際に一度 MAC-Layer Module へパケットを転送し MAC レイヤで行われる衝突回避の処理時間待機する。この衝突回避処理には、Request-To-Send(RTS) や Clear-To-Send(CTS)、Distributed Coordination Function Interframe Space(DIFS) などがある。MAC レイヤのエミュレーションは、MobiNet の大きな特徴であるが、無線通信における MAC レイヤ上での通信を再現しているわけではない。MobiNet が再現するのは RTS-CTS-Data-ACK のための MAC モデルであり、これによってパケットの送信時間を制御する。そのため、無線通信を再現するのではなく、無線通信における送信時間の再現を行っている。

MobiNet は、アドホックネットワークに対応させるために、ルーティングデーモンとの連携が必要となる。MobiNet はルーティングプログラムから各 VN へのパス情報を取得する。そして、IPv4 パケット受信時に送信元 IPv4 アドレスと送信先 IPv4 アドレスのペアから経路の探索を行う。このような動作モデルから、MobiNet はルーティングプログラムごとに個別のモジュールを実装する必要がある。

### Mininet

Mininet は、1 台のコンピュータ上で複数のリンク、スイッチから構成される仮想ネットワークを構築するエミュレータである。Mininet は、OpenFlow や SDN を使用した開発を対象とし、Openflow ベースのネットワークコントローラを使用する。

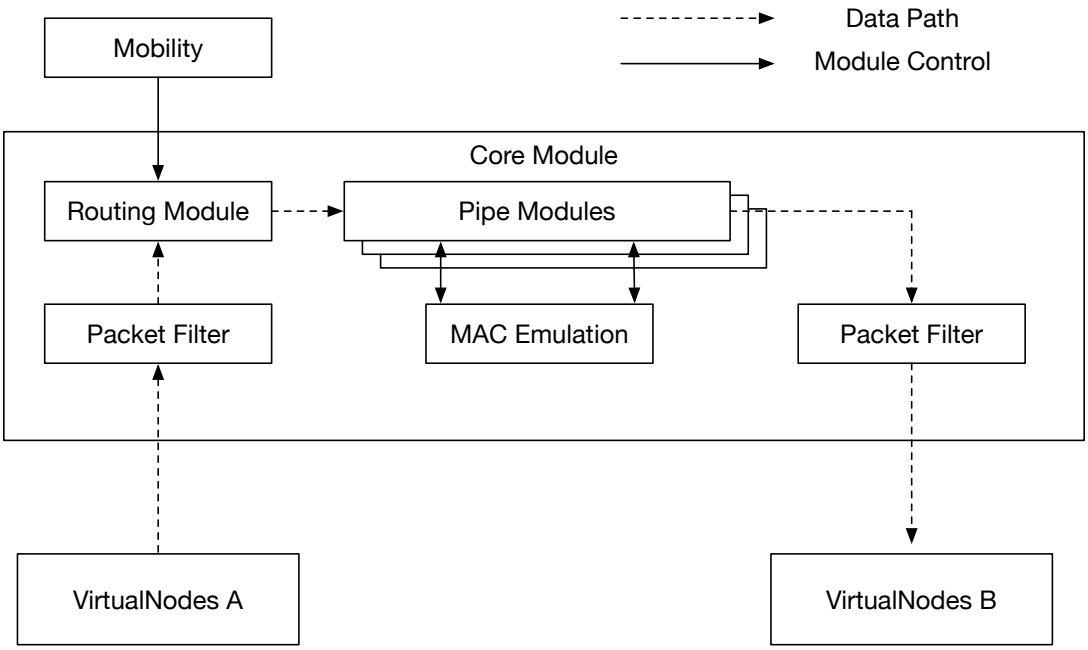



図 2.7: MobiNet モジュールによる伝搬エミュレーション

Mininet によって構築されるネットワーク構成は図 2.8 のようになる。Mininet はプロセスベースの仮想化によるホストと namespace を用いた仮想ネットワークを作成する。Mininet で作成されるホストは、namespace で分離されたプロセス空間で動作し、Linux で動作するアプリケーションを改変することなくそのまま利用できる。そのため、ホスト上でルーティングデーモンを動作させればルータ、サーバソフトウェアを動作させればサーバとして動作する。また、各ホストが持つネットワークインターフェースに、QoS の機能を適用することで、帯域幅や遅延時間の制御が可能である。Mininet で作成されるネットワークスイッチは、ホスト間にリンク毎に分離され、OpenFlow によるフローベース制御が行われる。

Mininet に無線ネットワーク機能を追加したエミュレータとして Mininet-WiFi [20] がある。Mininet-WiFi は、Mininet 仮想 WiFi アクセスポイントと仮想 WiFi ステーションの機能と追加した実装である。Mininet-WiFi では、無線インタフェースのエミュレーションにハードウェアエミュレータを使用している。

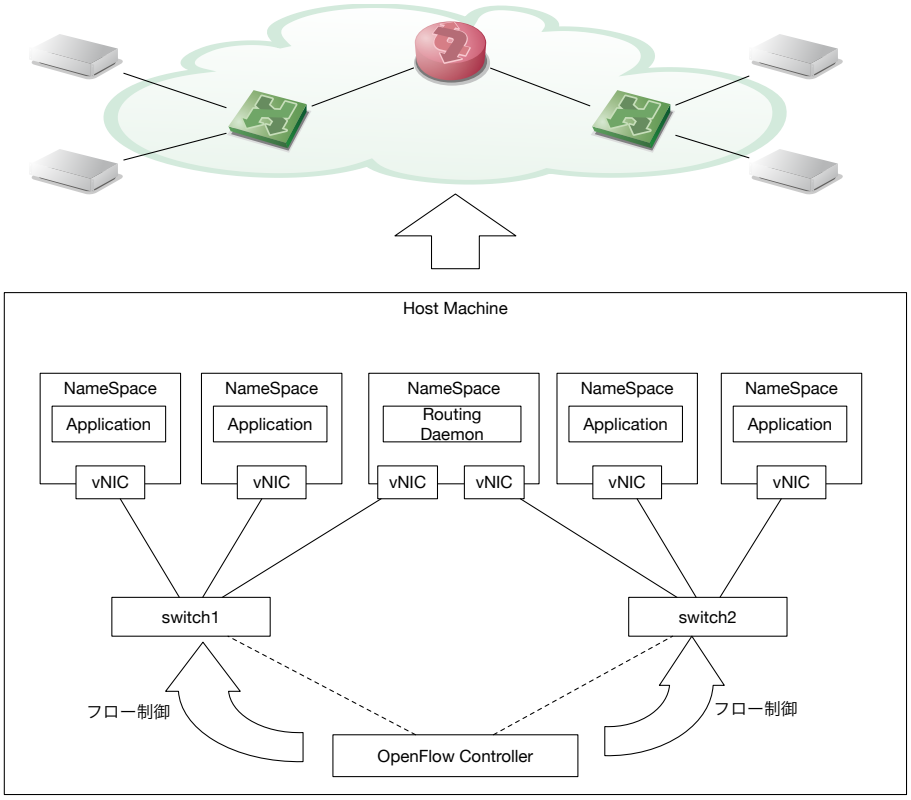



図 2.8: Mininet によるネットワーク構成

## 2.4　統合的なネットワーク技術検証基盤

2.3.1 節では、施設内で提供される無線ネットワーク環境を用いたネットワークテストベッドについて述べたが、これに対し有線ネットワーク上に無線ネットワークを再現しアプリケーションソフトウェアの検証基盤として利用可能な研究提案も存在する。本節では、このようなソフトウェアによる手法について述べる。

### 2.4.1　SHIVA: 耐災害 ICT 統合検証プラットフォーム

SHIVA [21, 22] は、マルチホップ無線技術や、ISP (Internet Service Provider) ネットワーク、車々間通信、そしてユーザが使用するアプリケーションなどを動作させ、実際のネットワーク環境に近い状態を再現すること可能な耐災害 ICT 統合検証プラットフォームである。そして、震災の発生をイベントとしてネットワークの障害から復旧までの変化

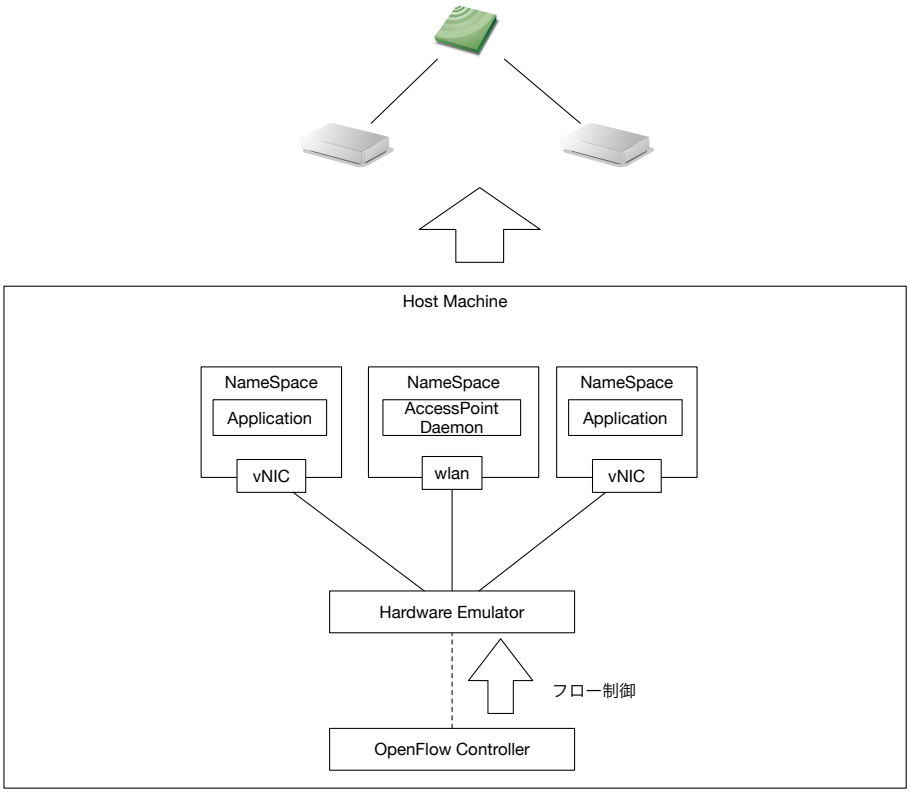



図 2.9: Mininet-Wifi によるネットワーク構成

を再現できる。SHIVA によって構築したネットワークには、実際のインターネット上で使用されている技術を用いているため、震災時のネットワーク障害、耐故障性の検証が可能である。

通常、ISP は安定した有線ネットワークを用いて通信路をユーザへ提供しているが、大規模な災害が発生した場合、この有線ネットワークが切断される可能性がある。SHIVA では、切断された有線ネットワークの代替手段として、長距離無線ネットワーク技術や Delay/Disruption Tolerant Networking(DTN) [23] を用いたデータ配送を用いて震災時の通信インフラを構築する。このような状況を、実際に構築することは不可能であるが、であるからといって、検証の不十分な技術を緊急時のインフラとして利用するのは信頼性に問題がある。そこで、SHIVA を用いることで、震災前のネットワーク状態から震災後の状態への変化や、長距離無線ネットワークや DTN を用いた無線メッシュネットワークの構築を行うことで緊急時に導入が想定されるネットワーク技術の検証が可能となる。

### 2.4.2 Network Emulation and Realtime Visualization Framework(NERVF)

NERVF は、SHIVA をより発展させた統合検証環境である。SHIVA では、ISP ネットワーク、無線メッシュネットワーク、車々間通信から構築されたインターネット環境を再現し、耐災害を想定した統合検証環境であったのに対し、NERVF はインタラクティブな無線メッシュネットワーク実験を主眼においている。

NERVF が提供するインタラクティブな無線メッシュネットワークとは、ユーザが統合検証環境を操作し、その結果が無線メッシュネットワークに反映されることを指す。ユーザは、NERVF 環境内で WiFi 中継機であるタワーや Unmanned aerial vehicle(UAV) の設置や操作を行うことである程度無線メッシュネットワークの状態を制御することができる。また、NERVF の環境ではトラフィックを送受信するノードが設置されており、ユーザが配置したタワー、UAV、そして自動的に移動する自動車間で構築された無線メッシュネットワーク上で通信を行う。無線メッシュネットワークの状態は、3D で可視化されており、タワーと UAV の配置によってどのようにネットワークの状態が変化するかが一目で把握できるようになっている。

## 2.5 既存の無線ネットワークエミュレーションの問題点

これまでに述べた既存研究の手法は、ある特定の状況や通信メディアを用いた通信に限定されているものや大規模に展開することが難しいものが多く、使用可能なネットワーク技術に制限がある。図 2.10 は、既存研究の手法を Fidelity と Scalability の観点から分類を行った図である。無線テストベッドや FPGA を用いた伝搬エミュレーションは、高い忠実度を持つ反面、規模追従性に関しては 100 ノードの程度となっている。一方で、伝搬エミュレータは伝搬エミュレーションをネットワーク層で行っていることから、無線ノードの識別や伝搬特性の適用を行うためにネットワーク層のヘッダ情報を必要とする。そのため、経路情報の交換を行うメッセージにネットワーク層のヘッダが付与されない B.A.T.M.A.N-adv [24] のような、データリンク層で動作するソフトウェアを動作さ

せた際に伝搬特性を適用することが出来ない。また、ネットワーク層で動作するルーティングプロトコルにおいても、同一ネットワーク内でマルチホップさせた場合に特別な処理を行う必要がある。QOMET は、OLSR を用いたアドホックネットワークの構築が可能であるが、送信元 IPv4 アドレスと送信先 IPv4 アドレスからノード間の遅延、帯域幅、パケットロスを適用するため、アドホックネットワークで発生するマルチホップ通信時に問題が発生する。そこで、QOMET は自身のルーティングテーブルを参照し、送信先アドレス宛のネクストホップアドレスを取得する。そして、ネクストホップアドレス宛の遅延時間と帯域幅、パケットロス率を送信先アドレスの伝搬特性へ上書きする。この方法はルーティングテーブルを参照するためネットワーク層のプロトコルに依存している。そのため、伝搬エミュレータの実装後に新たに提案されたネットワーク層のプロトコルに対応するためには、ネットワークエミュレータに大幅な変更が必要となる。このように、ネットワーク層のプロトコルに依存した伝搬エミュレータは、新しいネットワークプロトコルへの対応するためのコストが大きい問題がある。

ヘテロジニアスなネットワーク環境では、多数の無線デバイスが動作し、様々な無線技術、ネットワークプロトコルが混在して動作している。このネットワーク環境を再現し、その上でソフトウェア検証を可能とするためには、高い忠実度と規模追従性を併せ持つエミュレーション機構が必要である。

### 2.5.1　多様化するネットワーク技術への対応

施設として構築されたネットワークテストベッドは、ハードウェアを用いて高忠実なネットワーク環境の上で検証が可能であるが、この忠実性はハードウェアの性能に依存している。そのため、新しい無線技術が提案された場合、既設のハードウェアの変更が必要となるが、これには金銭的コストが大きな負担となってしまう。ソフトウェアベースで実装しているハードウェアエミュレータや Mininet-Wifi では、実際に無線技術、機能を利用可能であるが、単一コンピュータ上でのみ動作する。一方で、統合的なネットワーク技術検証基盤は無線ネットワークの中でも特定の状況を想定した設計、実装となっている。

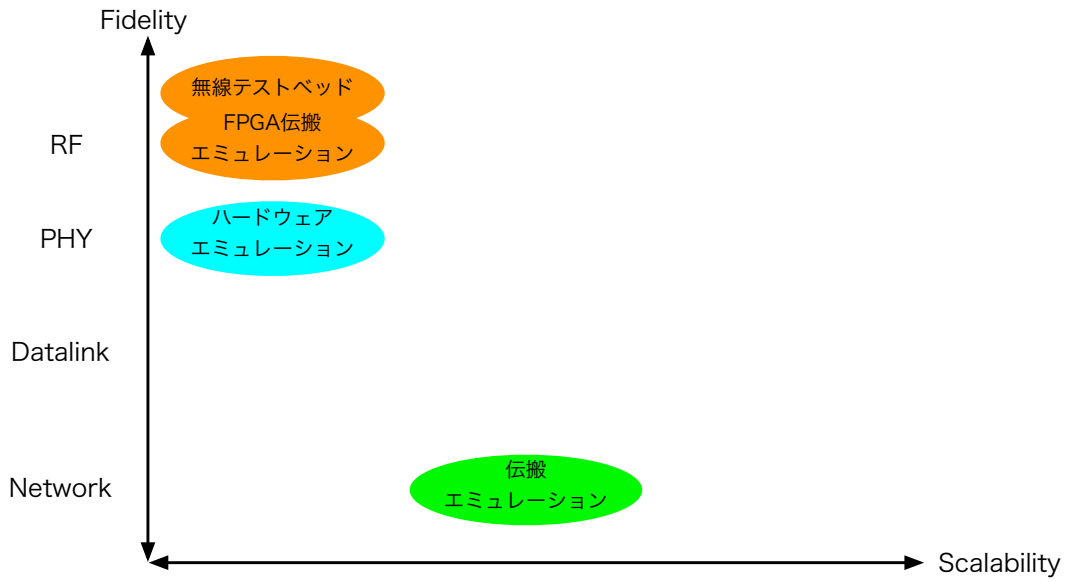



図 2.10: 既存研究の分類

そのため、新しいネットワークプロトコルや異なるレイヤで実装されたルーティングプロトコルへの対応ができないことが挙げられる。SHIVA や NERVF は、Optimized Link State Routing Protocol(OLSR) [25] を使用して無線メッシュネットワークを構築しているが、OLSR が唯一の無線メッシュネットワーク技術ではない。無線メッシュネットワーク技術には、想定規模や無線ノードの種類に違いがあるが、様々な研究が行われ、実装も進んでいる。ネットワークテストベッドとして、ヒトが持つデバイスや街に設置されるであろうデバイスを再現し検証を行うためには、様々な技術に対応可能な無線ネットワークエミュレーションが可能でなければならない。

### 2.5.2　ネットワークエミュレータの規模追従性

施設として構築されたネットワークテストベッドは、実際に無線インターフェイスを持つハードウェアを用いているため、電波干渉を回避、減衰を発生させるために無線ノード間の距離が必要となる。そのため、電波の送受信を行っている限りどのような方法と採ったとしてもある程度の広さがなければならず、規模追従性は乏しい。この無線ノード間の距離を保ったまま大規模な無線ネットワークを構築することは難しいだろう。一方で、統

合的なネットワーク技術検証基盤はソフトウェアでの無線エミュレーションを行っているため、規模追従性は無線ネットワークエミュレーションに使用しているエミュレータに依存する。SHIVA や NERVF は、無線ネットワーク内に 100 台程度の移動無線端末を動作させることが可能であるが、ヘテロジニアスな無線ネットワークでは、様々な端末、通信メディアが混在する。例えば、モバイル端末が使用する 3G や LTE 用基地局であるマイクロセルが 1 km から 3 km 辺り最大 256 人のユーザを収容している。そのため、ヘテロジニアスネットワークを構築するためには、1,000 ノード以上が存在する無線ネットワーク環境が必要であると考えられる。

## 2.6　無線ネットワークエミュレーション基盤に対する本研究の取り組み

ネットワークテストベッド上で多様化する情報サービスの検証を十分に行うには、様々なネットワークが導入されたインフラ基盤を再現する必要がある。しかし、これまで述べたように既存の無線ネットワークエミュレーション手法では、新規に提案、導入された技術への対応が困難であり、また規模追従性に関しても問題がある。そこで本研究では、図 2.11 に示すように、既存研究ではネットワーク層で動作していた伝搬エミュレーションを、忠実度と規模追従性の向上を実現した。そして、ハードウェアエミュレーションに対して複数ノード間での無線通信を可能とすることで規模追従性を持たせた。これらの拡張によって、ネットワーク機器やプロトコルに依存しない伝搬エミュレーションと大規模な無線通信を有線ネットワークで実現することを可能とした。

本論では、ネットワークテストベッド上に多種多様な無線ネットワーク環境を大規模に再現する無線ネットワークエミュレーション機構 NETorium を提案し、その設計と実装について議論する。NETorium は、無線ノード間で発生する遅延時間や可用帯域などの伝搬特性を再現する伝搬エミュレータ Meteor と有線ネットワーク上で無線インターフェイスを用いた通信を実現する Asteroid から構成される。既存の無線ネットワークエミュレーション機構は、特定のルーティングプロトコルやネットワークプロトコルを対象とし

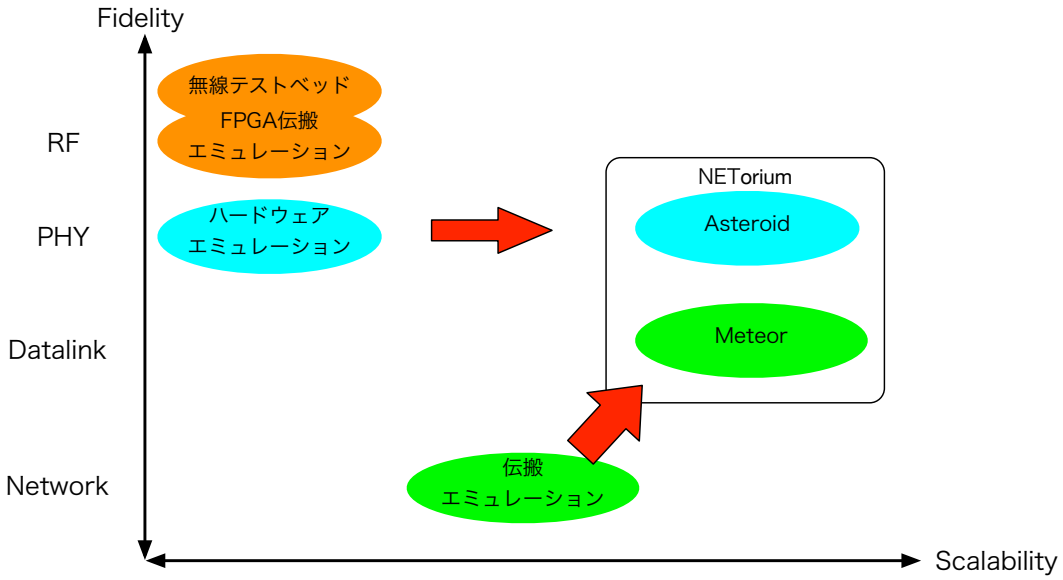



図 2.11: ヘテロジニアスネットワークエミュレーション手法への拡張

ているため、新しい技術の導入が困難となっている。NETorium では、ネットワーク機器やプロトコルに依存しないエミュレーション機構と持つことで、物理機器を含めたネットワークの構築や様々なレイヤで動作するソフトウェアの検証が可能である。また、有線ネットワーク上での仮想的な無線通信が可能であるため、無線規格で定義されている認証やメッシュプロトコルなどの技術を用いることが可能である。

# 第 3 章

# ヘテロジニアスネットワークの再現

本章では、ヘテロジニアスネットワークを再現するために必要となる要素技術を整理し、ネットワークテストベッドが進化するネットワーク技術に追従していくためにはどのようなエミュレーションを行わなければならないかについて議論する。そして、ヘテロジニアスネットワークを再現するアーキテクチャについて述べる。

## 3.1　ヘテロジニアスネットワークの構成する要素技術

本節では、ヘテロジニアスネットワークを構成する要素技術として、ソフトウェアのみで実現されているネットワークプロトコル技術と制御プロトコルを含めた物理媒体に紐づいている通信技術の 2 側面から整理を行う。そして、各要素技術において無線ネットワークエミュレーションを行うための課題を明確にする。

### 3.1.1　様々な箇所で動作するネットワークプロトコル技術

ネットワーク技術は、基本概念である OSI 参照モデルに従って設計・実装が行われている。そして、OSI 参照モデルで定義されている各層では、新しい技術が導入されたとしても上位層と下位層とのインタフェース整合性がとれていれば通信が可能となる。ここでは、OSI 参照モデルで物理媒体を含まず、ソフトウェアのみで実現されている技術について述べる。

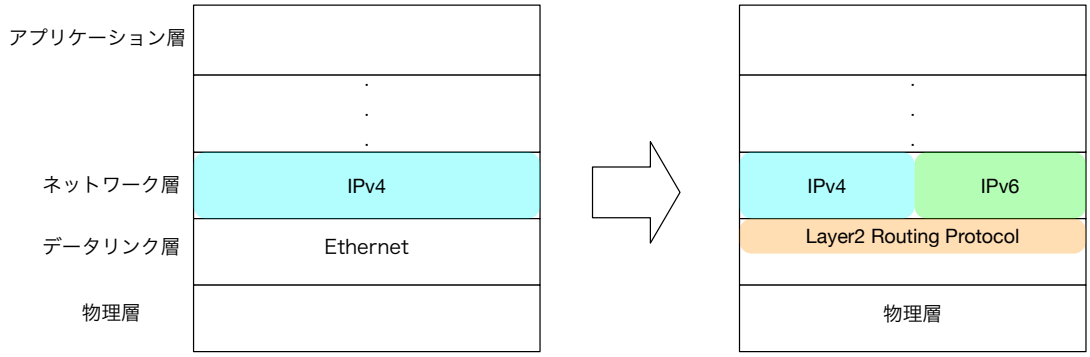



図 3.1: 増加するネットワークプロトコル

OSI 参照モデルにおいて、コンピュータ間の疎通製はネットワーク層で提供されている。ネットワーク層で利用される技術として、 Open Shortest Path Fast(OSPF) や Border Gateway Protocol(BGP) のような経路情報を交換し、インターネット上のホストへパケットを送受信可能とする技術である。これは、無線ネットワークでも変わらず、経路計算に使用されるパラメータ、計算式は異なるものの、ホスト間の到達性を実現しているのはネットワーク層である。しかし、様々な技術革新が行われた結果、この通信路の構築方法に大きな変化が起きている。

IPv4 アドレスが枯渇し、新しい IPv4 アドレスの配布が不可能となったため、新しいネットワーク層のプロトコルである IPv6 への移行が望まれている。これまでのルーティングプロトコルは、IPv4 のみを対象としていたため、IPv6 の経路交換には新しいアプリケーションが用いられる。また、フラットなレイヤ 2 ネットワークを構築するための技術としてレイヤ 2 ルーティングプロトコルなども提案されている。その結果、図 3.1 に示すように、ホスト間の通信を実現するネットワークプロトコルが OSI 参照モデルの層を跨いだ形で実現されている。

無線ネットワークエミュレーションは、無線ネットワークにおけるホスト間の振る舞いを再現することで、アプリケーションの検証を行うことが目的である。

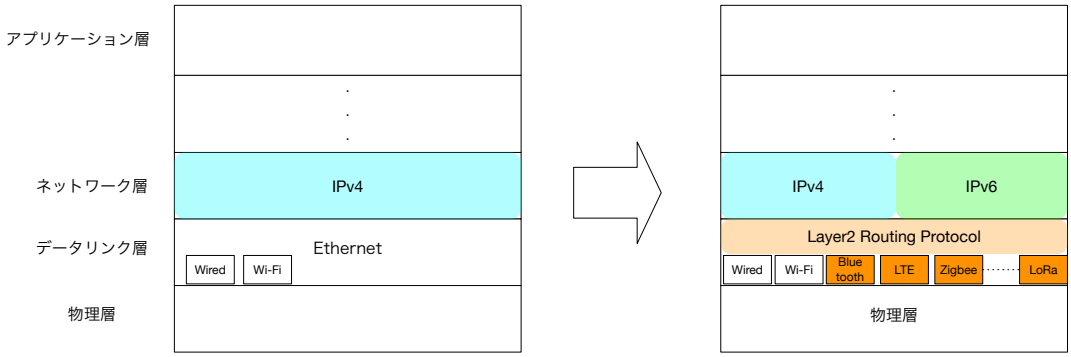



図 3.2: MAC 層に混在する多種多様な通信メディア

### 3.1.2　多種多様な通信メディア

多様化するネットワーク技術には、ネットワーク層、データリンク層だけではなく、通信メディアの多様化も挙げられる。無線ネットワークの通信メディアは、それぞれが規格化された技術を用いており、OSI 参照モデルとは異なるプロトコルスタックを持っていることがある。図 3.2 は、例えば、Wi-Fi (IEEE802.11) は、OSI 参照モデルの物理層からデータリンク層の MAC 複層までで動作するが、Bluetooth は IEEE 802.15.1 としてまったく異なるプロトコルスタックを持っている。これらの通信技術は、認証や、機能セット、使用する周波数帯などの情報交換、コネクション管理を独自に持っている。そのため、既存手法のような伝搬エミュレーションでは、遅延時間や帯域幅などの再現は行えたとしても、通信メディアが持つ機能をアプリケーションに提供することができない。

このような通信メディアをテストベッド上で通信可能とするためには、通信メディアから送信されるデータを有線ネットワーク上で配送する仕組みを実現する必要がある。

## 3.2　有線ネットワーク上でのヘテロジニアスネットワーク

これまで無線ネットワーク技術の多様化について述べた。本論文では、ネットワークテストベッドがこれらの技術に対応し、検証基盤として成り立つために必要となる要素をプロトコルへの対応と通信メディアの対応の 2 つに分離した。本節では、2 つの問題の根本

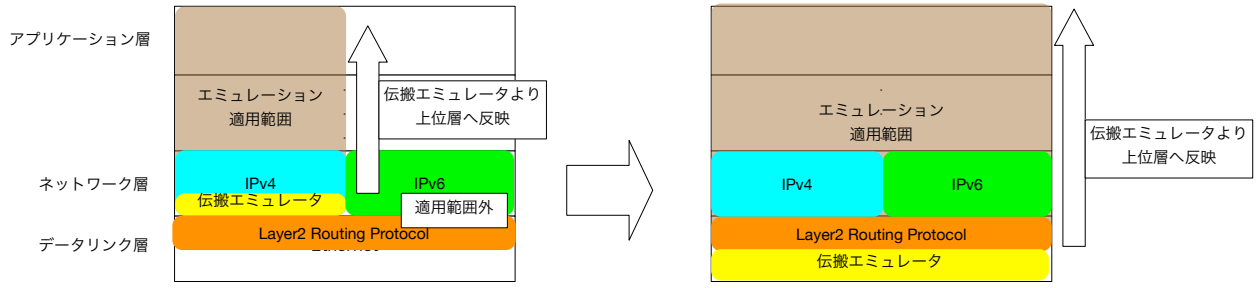



図 3.3: プロトコル非依存な伝搬エミュレーションを可能とする適用レイヤ

的な原因とその解決策について述べる。

### 3.2.1　プロトコル非依存性

これまでの伝搬エミュレータは、特定のネットワークプロトコル、アプリケーションソフトウェアを対象としているため、OSI 参照モデルにおけるネットワーク層での動作を設計を行い、その上で無線通信にて発生する伝搬特性、送受信のタイミング制御の再現をおこなっている。しかしながら、ネットワーク技術の発展によって、新しいネットワーク層のプロトコル、ネットワーク層よりも低いレイヤで動作するプロトコルが新規に提案されている。これまでのネットワーク層の特定のプロトコルに依存した設計では、全ての機能に対応できない状況となっている。これらの技術に対応していくためには、伝搬エミュレータは、プロトコル非依存な伝搬エミュレーションを行える設計でなければならない。

プロトコル非依存な伝搬エミュレーションを行う方法として、既存の伝搬エミュレータが動作しているレイヤよりも下位に位置するレイヤで伝搬エミュレーションと行う方法が考えられる。図 3.3 に示すように、これまでの伝搬エミュレータでは、ネットワーク層の IPv4 のみを対象としているため、伝搬特性が反映されるのは、ネットワーク層より上位層の範囲かつ IPv4 を用いている場合に限られる。これを伝搬エミュレーションを有線ネットワークで用いられる中で最も下位となるレイヤで適用することで、上位層でどのようなプロトコルが用いられていたとしても伝搬特性が反映される。これにより、ネットワーク層のプロトコルに依存せずに伝搬特性を再現することが可能となる。

### 3.2.2 ヘテロジニアスな通信メディアの制御

ネットワークテストベッドで、通信メディアまでの模倣した検証を行うためには、Orbit のような実デバイスを用いたネットワークテストベッドを構築するか、FPGA などを用いて検証対象となるアプリケーションを動作させる OS に無線インタフェースを提供する必要がある。しかし、これらの方法は、各々の論文で規模追従性に限界があることが述べられている。通信メディアを含めた大規模な無線ネットワーク上での検証を行うためには、ソフトウェアベースで構築可能なエミュレーション手法でなければならない。

ソフトウェアベースで多種多様な通信メディアを仮想的に作成するハードウェアエミュレータは数多く存在するが、構築手法が 1 台のホスト上でコンテナ技術などを用いて構築する手法となっている。1 台のホスト上で複数のコンテナを作成し、無線ネットワークを構築した場合、その環境の規模追従性はホスト及び OS の性能に制限される。これらの手法が 1 台のホストに制限される原因は、データリンク層のデータ形式が有線ネットワークと異なるため、有線ネットワーク上に無線フレームを送信することができないためである。ソフトウェアベースかつ、大規模な無線ネットワーク環境を実現するためには、仮想化された無線インタフェースが送受信する無線フレームを有線ネットワーク上で扱えるようにする必要があるが、これを実現するエミュレーションは未だ存在しない。

本論文では、有線ネットワーク上に仮想的な無線ネットワークを構築するエミュレータを仮想無線ネットワークエミュレータと呼ぶ。仮想無線ネットワークエミュレータは、図 3.4 に示すように、様々な無線インタフェースが送受信する無線フレームを解析し、有線ネットワーク上に送信可能なフォーマットへ適切に変換することで、ホスト間での無線通信を提供する。これにより、従来のソフトウェアベースで実現していたエミュレーション手法の問題であったホストの性能による制限を取り除くことが可能となる。

そして、有線ネットワーク上で送受信可能なフォーマットへ変換した後に、伝搬エミュレーションを適用することでヘテロジニアスな無線ネットワークをネットワークテストベッド上に構築することが可能となる。

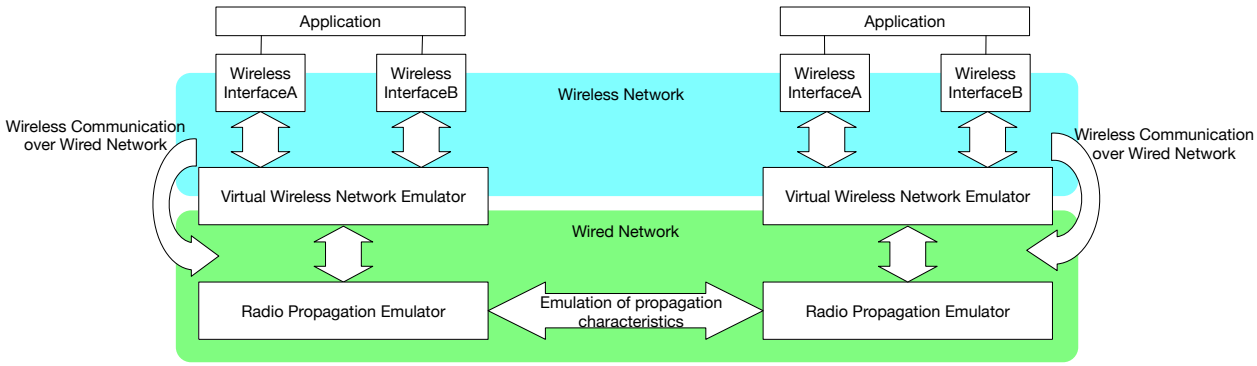



図 3.4: ネットワークテストベッド上でのヘテロジニアスネットワークの再現

# 第 4 章

# Meteor

## 4.1 はじめに

ネットワークテストベッドは、新たに開発したアプリケーションソフトウェアや情報サービスの検証を行うために構築された実験環境である。ネットワークテストベッドには、PlanetLab [26] のようにインターネット上にオーバーレイネットワークを構築するものと StarBED [27, 28] や ORBIT のように施設内に閉じた環境で閉鎖した実験ネットワークを構築するものがある。オーバーレイネットワーク上で実験を行う場合、検証の対象となるアプリケーションソフトウェアや情報サービスを動作させるサーバは、インターネット上に展開されるため、他のユーザのトラフィックや他の拠点のサーバとの間で発生する遅延時間や帯域幅の制限のような外部からの影響があるネットワーク環境での検証が可能である。これに対し、StarBED や ORBIT のような施設内で実験を行う場合、検証環境には検証を行うサーバとネットワーク機器のみが存在するため、他のユーザのトラフィックの影響を受けず、低遅延・広帯域な環境で検証が可能である。StarBED は、多数のサーバが有線ネットワークで接続されたクラスタ環境となっているため、他のユーザのトラフィックの影響がないことに加えて、低遅延、広帯域な環境での検証を行うことができる。このような環境は、アプリケーションソフトウェアが送受信するトラフィックへ影響する要素が少ないため、同じ構成、条件での実験を繰り返し行うことが可能である。一方で、無線ネットワークのような遅延や帯域幅、パケットロスが刻一刻と変動するよう

な通信環境を構築するためには、擬似的に無線空間を再現する必要がある。モバイル端末や IoT 端末のような無線を用いてインターネットへ接続する小型端末が増加している中、ネットワークテストベッド上に擬似的な無線空間を作り出し情報サービスの検証を行うことが可能な検証環境が求められている。

このような要望に対して、有線ネットワーク上に擬似的な無線空間を再現する伝搬エミュレータが数多く提案されている。伝搬エミュレータは、有線ネットワーク上に接続されたサーバ間で、無線空間で発生する遅延時間や帯域幅を再現することで、擬似的に無線空間を再現する。伝搬エミュレータを用いることで、無線環境での利用を想定しているアプリケーションソフトウェアの検証を繰り返し容易に行うことが可能となる。

既存研究にて提案されている伝搬エミュレータの代表的なものに、QOMET [29] や MobiNet [30] がある。これらの伝搬エミュレータは、特定のネットワークプロトコルを対象とし、無線環境で発生する遅延時間や可用帯域の揺れを再現することで、アプリケーションソフトウェアが無線ネットワークを使って通信しているかのように見せかけることができる。これまで無線ネットワークでの利用を想定しているアプリケーションソフトウェアの検証は、この伝搬エミュレータにより作られた擬似的な無線空間で十分対応できていた。しかしながら、新しい無線技術の提案により検証環境の再現が難しい場面が出てきている。

本章では、これらの問題を解決する伝搬エミュレータ Meteor の設計と実装について述べる。Meteor は、特定のネットワークプロトコルを対象とするのではなく、プロトコル非依存な伝搬エミュレータとして動作する。これにより、これまで検証が困難なものであった無線技術の検証が可能とする。

## 4.2 伝搬エミュレータ

### 4.2.1 ネットワークテストベッドで利用されるネットワークエミュレータ

ネットワークテストベッドには、ある特定の環境を想定したものや、汎用的に利用可能なよう低遅延・広帯域な有線ネットワークとそこに接続されたサーバクラスタで構成され

るものなどがある。Orbit は、無線アドホックネットワークに焦点を当てたネットワークテストベッドであり、前者にあたる。このようなネットワークテストベッドは現実性が高い反面、規模追従性の向上や新しい技術への対応に多大な金銭と労力を必要とする。これに対し、StarBED は多数の PC サーバを有線ネットワークに接続し、様々な OS やアプリケーションソフトウェアを検証可能なテストベッドであり後者にあたる。StarBED は、 4.1 節で述べたように、安定した検証環境で同じ条件で何度も繰り返し検証を行える。しかしながら、遅延時間や利用可能な帯域幅、パケットロス率といったネットワーク特性が周辺環境によって大きく変動する無線ネットワークのような環境を擬似的に再現する手法が必要となる。本節では、ネットワークテストベッドで利用可能な擬似的な無線空間を再現する手法について述べる。

## 4.3 既存の伝搬エミュレータの問題点

### 4.3.1 プロトコル依存な伝搬エミュレーション

既存の伝搬エミュレータである QOMET や MobiNet は、設計思想から特定のルーティングプロトコルやネットワークプロトコルに依存した伝搬エミュレーションを行っている。QOMET は、ネットワーク層のプロトコルの 1 つである IPv4 のみを対象としており、IPv6 には対応していない。また、無線アドホックネットワークに対応するため、QOMET を実行しているノードのルーティングテーブルを参照しネクストホップとなるノードの情報を取得する必要がある。そのため、QOMET が扱えるプロトコルは、IPv4 かつルーティングテーブルにネクストホップ情報が載る場合に限られる。

MobiNet は、無線アドホックネットワークを対象とした設計となっているため、ルーティングプログラムとの連携が必須である。そして、ルーティングプログラムとの連携を行うためは、MobiNet が持つルーティングプログラム毎のモジュールを実装する必要がある。また、MobiNet も QOMET 同様 IPv4 のみの対応となっており、IPv6 への対応は言及されていない。そのため、MobiNet で扱えるプロトコルは、IPv4 かつ MobiNet がサポートしているルーティングプロトコルである場合に限られる。

このように、既存の伝搬エミュレータは対象としているプロトコルに依存した設計および実装となっているため、新しいプロトコルへの対応が難しい。特に IPv4 は、アドレス枯渇の問題から IPv6 への移行が望まれており、将来 IPv4 アドレスを持たないネットワーク環境が構築される可能性もある。無線アドホックネットワークに関しても、IPv6 を使用する OLSR や、データリンク層でトポロジを構築する B.A.T.M.A.N-adv のようなルーティングプロトコルも提案されている。伝搬エミュレータは、擬似的に無線環境を作り出すソフトウェアであるため、その上で利用するアプリケーションソフトウェアを制限することは避けるべきである。伝搬エミュレータがこれらのプロトコルへも対応していくためには、特定のルーティングプロトコルやネットワークプロトコルに依存しない設計・実装でなければならない。

### 4.3.2 スケーラビリティ

既存研究で提案されている伝搬エミュレータは、数十台から多くとも 200 台程度の規模を想定している。QOMET は、実際に 100 台のノードを用いてアドホックネットワークを構築している実績がある [21, 22]。MobiNet は、論文中で 200 台のノードで動作させたと記述しているが、40 台での検証にとどまっている。しかし、IoT の技術を展開した世界では、1,000 台以上のノードが無線ネットワークに接続されることが予想される。少数のノードで構成されたネットワークでは、大規模ネットワークでのみ発生しうる問題を見逃す恐れがあるため、伝搬エミュレータは予想される規模を展開可能である必要がある。

## 4.4 プロトコル非依存な伝搬エミュレーション

これまでで、伝搬エミュレータの擬似的な無線空間の再現手法とその問題点について述べた。伝搬エミュレータが、今後提案されるであろうネットワーク層のプロトコルやルーティングプロトコルに対応していくためには、これらのプロトコルに依存しない伝搬エミュレータが必要である。本節では、これを実現する Meteor の設計概要について述べる。

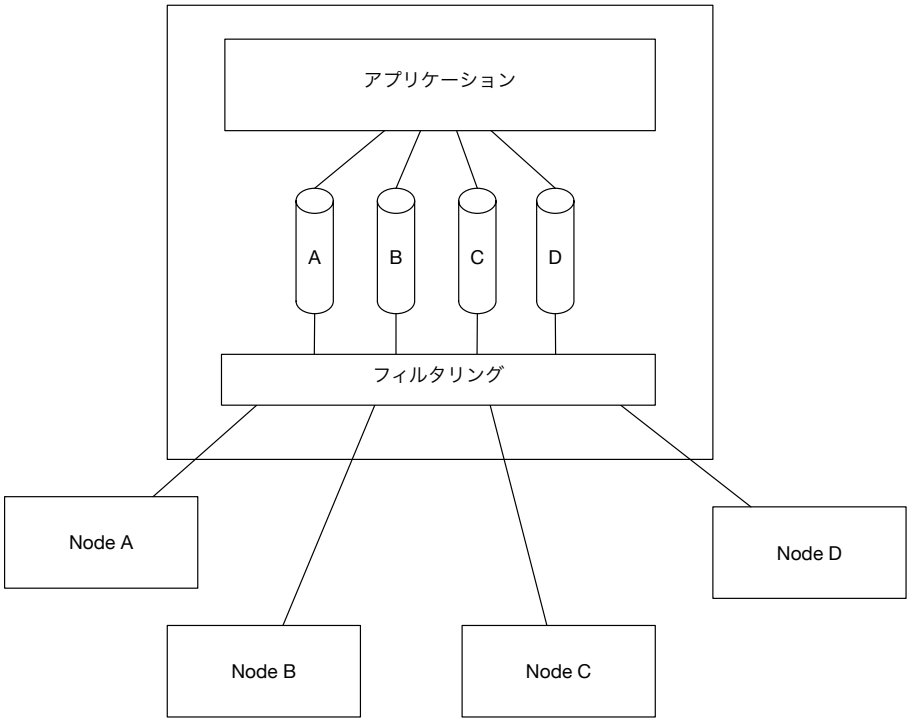



図 4.1: 伝搬エミュレータの制御

QOMET や MobiNet がネットワークプロトコルに依存し、ルーティングプロトコルへ対応するために複雑な処理を必要としている最も大きな理由は、ネットワーク層のプロトコルを対象とした設計となっていることである。伝搬エミュレータの動作は、図 4.1 に示すように受信したデータをフィルタリングし、各送信元、送信先アドレス毎に管理しているキューへ入れる。各キューは、伝搬エミュレータが管理している他のノード宛への遅延時間や帯域幅を管理する。このフィルタリングとキュー制御によって、伝搬エミュレータは無線空間を擬似的に再現しているため、フィルタリングにマッチしないプロトコルの通信は伝搬特性が適用されない。MobiNet は、ノード間の無線空間を再現する際に、MAC レイヤエミュレーションを行っているが、これは無線通信時に行われる衝突回避制御であり、フィルタリングは IPv4 パケットに対して行われるため問題点は変わらない。

この問題は、伝搬エミュレータが行っているフィルタリングをネットワーク層のプロトコルを対象にするではなく、データリンク層のプロトコルを対象とすることで解決可能であると考えられる。ネットワークテストベッド上では、ネットワーク層のプロトコルは

ユーザが自由に選ぶことが可能であるが、データリンク層のプロトコルはネットワーク機器の制御にも影響するため、自由に選ぶことは難しい。そのため、有線ネットワーク上で利用されているイーサネットを対象とすることで、有線ネットワーク上通信可能なプロトコルはほぼ全て対応可能である。

### 4.4.1 リンクエミュレータ

4.4 節にて、プロトコルに依存しない伝搬エミュレータを実現するためには、ネットワーク層ではなくデータリンク層を対象にしなければならないと述べた。これを実現するためには、伝搬エミュレータが使用するリンクエミュレータを再考する必要がある。本節では、伝搬エミュレータに利用可能なリンクエミュレータについて述べ、それぞれのメリットデメリットについて議論する。

#### ユーザ空間でのリンクエミュレータ

遅延挿入や帯域の制限を行うトラフィック制御は、サーバが送受信するトラフィックをネットワークバッファに一時的に滞留、または破棄を行うことで実現している。オペレーティング・システムのネットワークスタックはカーネル空間で実装されているため、これらの処理はカーネル内で行われる。そのため、ユーザ空間でリンクエミュレータを実装するためには、カーネル空間からユーザ空間へパケットを横取りする必要がある。ユーザ空間へのパケットの横取りは Netfilter nfqueue [31] や Divert Socket などを用いることにより実現できる。ユーザ空間へパケットを横取りした場合、カーネル空間で独自実装することと比較して実装が容易である。しかし、nfqueue や Divert Socket を用いた場合、カーネル空間からユーザ空間へのメモリコピーが発生するため、ここにオーバーヘッドが生じる。これらのパケット横取り手法を用いた伝搬エミュレータとして GINE [32, 33] が提案されている。GINE は、複数の仮想ルータをノード内に作成し、仮想ルータ間のトラフィック制御を行うエミュレータであるが、広帯域な回線をエミュレーションした際にスループットの低下が見られている。

#### カーネル空間でのリンクエミュレータ

カーネル空間で実装されているリンクエミュレータでは、ユーザ空間へのメモリコピーが発生しないため、オーバーヘッドは小さくなる。一方で、実装が難しくなり、カーネルのバージョンが変更された際に動作しなくなる可能性がある。そのため、カーネル空間で独自に実装を行ったリンクエミュレータを用いた伝搬エミュレータは少ない。Linux や FreeBSD、MacOS X では、OS 標準の機能としてリンクエミュレータが実装されており、既存研究で提案されている伝搬エミュレータはこれらを使用している。カーネル空間で動作する代表的なリンクエミュレータとして TC/Netem [34] や NISTNet [35]、Dummynet がある。これらのリンクエミュレータは Quality of Service(QoS) や、プロトコルテストのために実装されたものであり、多くの伝搬エミュレータはこれらの機能を柔軟に制御することにより無線空間の再現を行っている。これらは、各ディストリビューションでメンテナンスされており、ユーザ空間から API を操作することでカーネルのバージョンが変わったとしても制御可能である。

#### NISTNet

NISTNet は National Institute of Standards and Technology(NIST) が設計・実装していたリンクエミュレータである。NISTNet は ネットワーク層の入出力パケットをカーネルモジュールで横取りし、トラフィック制御を行う。NISTNet は Linux 2.6.14 で終了しており、現行の Linux では利用できない。

#### TC/Netem

TC/Netem は Linux で実装された広域ネットワークを再現するためのリンクエミュレーション機構である。Netem という名前は後述する遅延制御を行うモジュールであるが、Linux のトラフィック制御機構を総称して指すことが多い。以降、本論文では TC/Netem をトラフィック制御機構そのものを指し、Netem は遅延制御モジュールを指すものとする。

TC/Netem の制御用ユーティリティとして tc [36] が提供されている。TC/Netem はインタフェース毎の送信キューである Qdisc でイーサネットフレームを滞留、破棄することでトラフィック制御を実現している。Qdisc はネットワークインターフェースの送信キューであるためトラフィック制御を行う単位はパケットではなくフレームとなる。また、基本的に送信フレームのみ制御可能であり、受信パケットを扱うためには受信用のモジュールと Intermediate Functional Block device (IFB) インタフェースを使用する。

TC/Netem は、遅延挿入や帯域制限以外にもフレーム重複やリオーダーなどの機能を持っており、各機能はモジュールとして提供されている。各モジュールは、それぞれ自身で Qdisc を持ち、各 Qdisc は木構造で管理される。それぞれの Qdisc を連結し、様々なモジュールを組み合わせることで柔軟な制御が可能である。

図 4.2 は、TC/Netem のモジュールを連結させた場合の動作を示した図である。Qdisc は、16bit の ID を持ち、Qdisc を連結する場合は親の Qdisc の ID と子となる Qdisc の ID を指定する。Qdisc のモジュールには遅延挿入、パケットロスのための Netem、帯域制限を行うための Token Bucket Filter(TBF) や Class Base Queuing(CBQ)、Hierarchical Token Bucket(HTB) などがある。また、Qdisc にはクラスフルとクラスレスの概念があり、クラスフルなモジュールはトラフィックをクラス別にフィルタリングし各クラスでトラフィック制御を行う。Netem や TBF、ingress はクラスレスであり、CBQ と HTB はクラスフルとなる。クラスフルな Qdisc をフィルタリングするには、MAC アドレスや IP アドレス、ポート番号などを指定し転送先の ID を指定する。

### Dummynet

Dummynet は、Luigi Rizzo が提案したネットワークエミュレーション機構である。Dummynet は IPFW Firewall のモジュールとして提供されており、FreeBSD、Mac OS、Linux、Windows で利用可能である。図 4.3 は、パケットが Dummynet を経由する際の処理の流れを示している。IPFW は、プロトコルスタック内でデータリンク層の入出力を行う ether_demux、ether_output_frame、 ブリッジ処理を行う bdg_forward、ネットワーク層の入出力を行う ip_input、ip_output のいずれかで IPFW の処理に渡される。

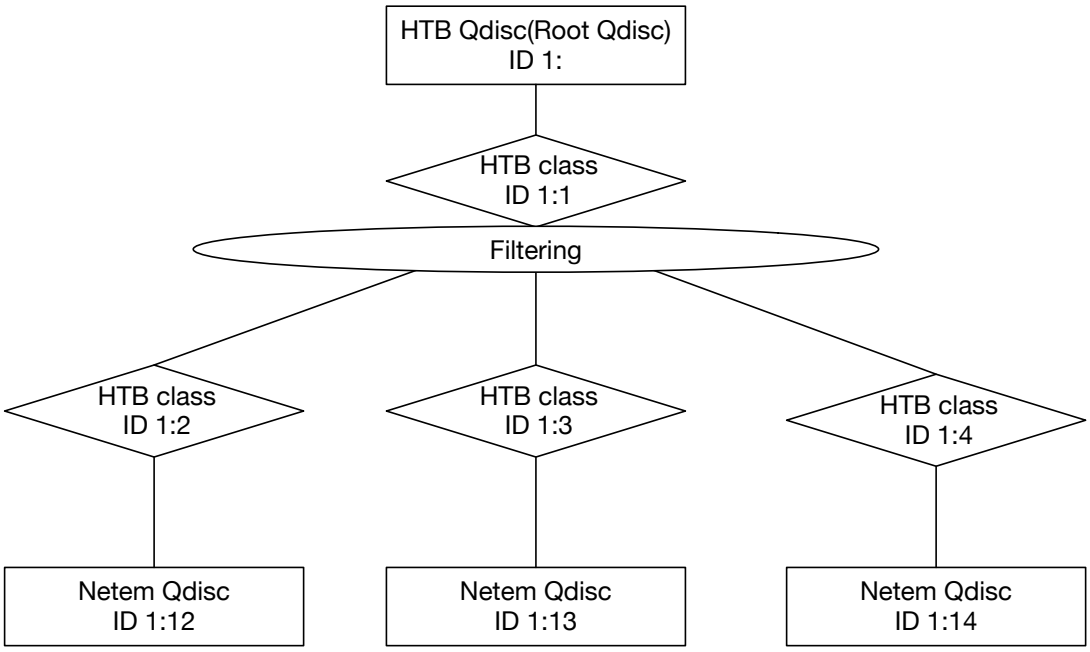



図 4.2: TC/Netem の動作

Dummynet は TC/Netem とは異なり、単体で遅延生成と帯域制限の機能を持っているが、遅延の値や帯域の制限値に固定値しか指定できない。これをランダムに変更したい場合、複数のフィルタルールを設定し、確率的にマッチさせることでジッタの再現を実現している。また、フィルタリングの識別子として、 IP アドレスのみの指定となっている。

### 4.4.2 フィルタリング

伝搬エミュレータが扱うプロトコルを変更することで、伝搬エミュレータの操作性を著しく変えてしまうことや、過去の実験データが使えなくなってしまうことにも問題がある。ネットワーク実験において、過去に使用したシナリオを用いて再実験を行うことは珍しいことではない。その際、伝搬エミュレータに機能が変更されたためにシナリオの作り直しが発生してしまうことはネットワークテストベッドの特徴である再現性を損ねてしまう。データリンク層に対応した伝搬エミュレータを使用するとしても、過去のシナリオは利用可能であるべきである。

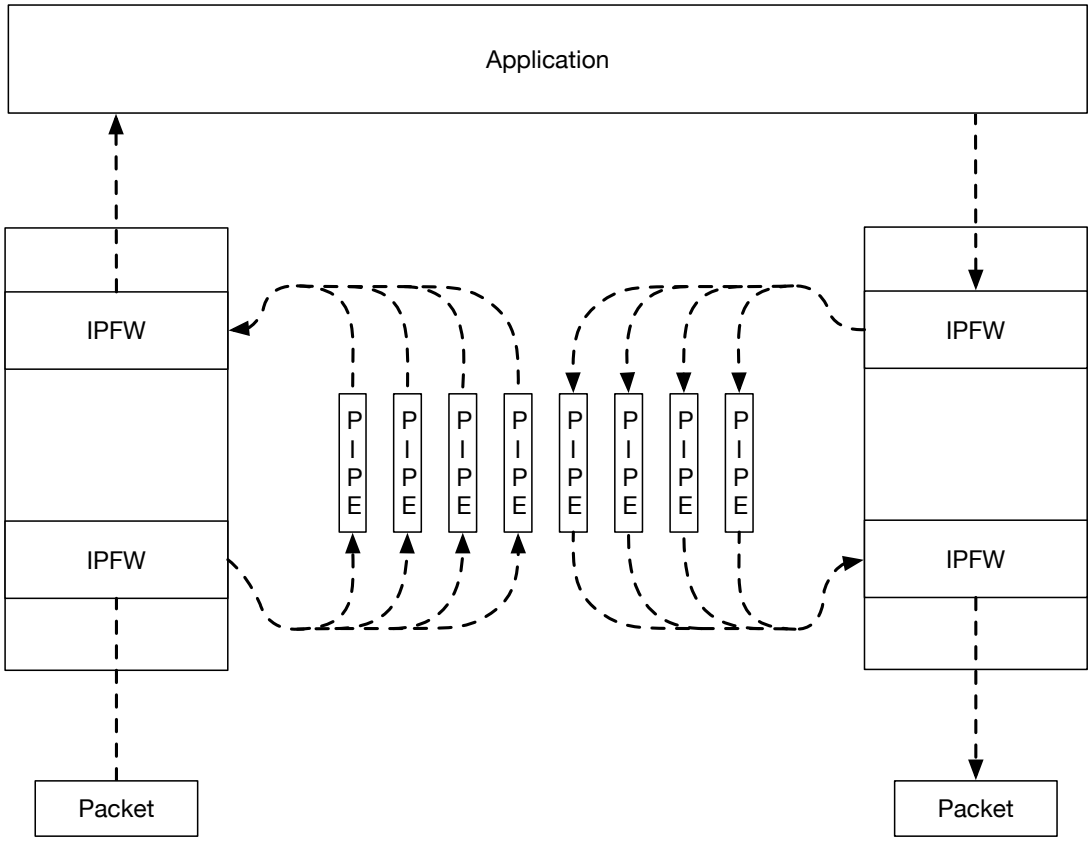



図 4.3: Dummynet の処理

### 4.4.3 タイマ制御

Meteor は汎用計算機、汎用 OS での動作を想定しているため、リアルタイム性が保証されない。しかし、伝搬ネットワークエミュレータは、頻繁に伝搬パラメータを変化させる必要があるため、ある時刻から実際に伝搬パラメータの変更処理が始まるまでの時間はマイクロ秒オーダであることが望ましい。伝搬ネットワークエミュレータにおけるタイマ制御は、構築可能なネットワーク構築可能なネットワークの規模に影響するものではないものの、指定時刻から大幅に遅れて設定の変更が行われてはエミュレータの精度に影響が出てしまう。また、伝搬パラメータの変更処理に必要な時間も考慮する必要がある。図 4.4 は、3 ノードとの間の遅延時間を変更した際の処理を表した図である。

無線空間の遅延時間や帯域幅の変動を忠実に作り出すには、高精度なタイマが求めら

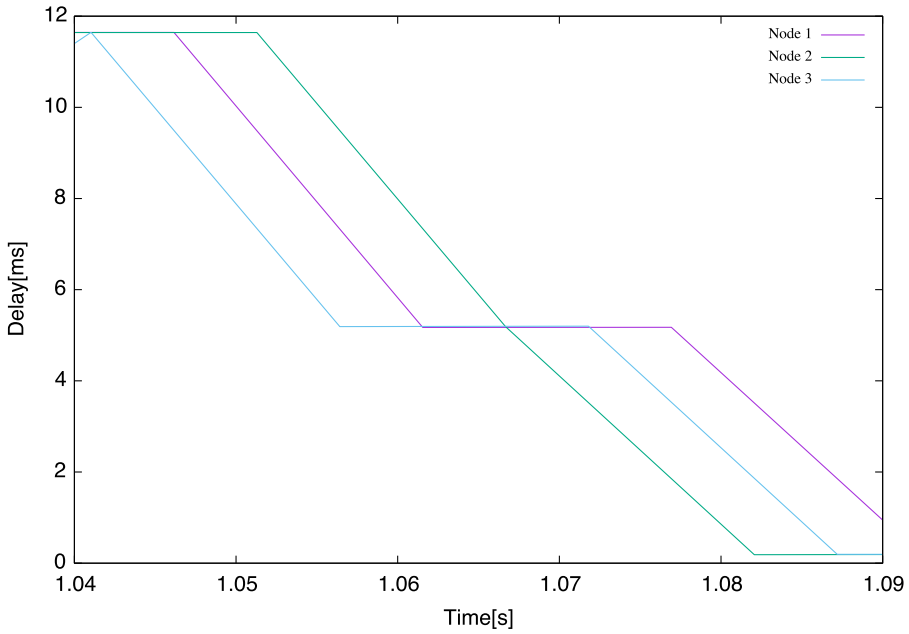



図 4.4: 各リンクへの伝搬パラメータの変更タイミング

れる。

## 4.5 Meteor の実装

ネットワークエミュレータの設計と実装に関する既存研究は多数存在するが、その多くがリンクエミュレータと呼ばれる機構を利用している。例えば QOMET では、遅延挿入や帯域制限の実現に Dummynet [37] を用いている。リンクエミュレータには、ソフトウェアとハードウェアが存在する。本章では、既存研究で多く使用されている代表的なリンクエミュレータおよび実装手法について解説を行う。

### 4.5.1 要件

既存研究で提案されているネットワークエミュレータは、ネットワーク層で動作するため、ノードの識別子に IPv4 アドレスを使用している。ネットワーク層で定義されている IPv4 アドレスは、ネットワーク内でユニークな識別子であるため、パケットからは送信元と送信先のノードしか判別できない。そのため、パケットをルーティングする実験で

は、次の転送先の情報をパケット以外から取得しエミュレーションの設定を変更する必要がある。

既存研究で提案されているネットワークエミュレータは、ノードの経路表を参照・変更し、その情報を元に動作する手法を取っている。しかし、ノードの経路表からはネットワーク層の情報しか得られず、既存研究で提案されているネットワークエミュレータは IPv4 にしか対応していないものが多い。本来、次の転送先を決定した後に書き換えられる情報はデータリンク層の識別子である MAC アドレスであり、ネットワーク層の識別子である IP アドレスは変更されない。

ネットワーク層に依存しないネットワークエミュレータを作るためには、各ノード間の遅延挿入や帯域制限を行うリンクエミュレータがネットワーク層よりも下位層で動作する必要がある。ネットワークテストベッドの有線ネットワークを変更することは物理構成を変更することとなるため、既存の構成を崩す事となる。そのため、Meteor で利用するリンクエミュレータはネットワーク層と物理層の間に位置するデータリンク層で遅延挿入や帯域制限が行えることが条件となる。そして、リンクエミュレータがデータリンク層で動作するためには、ノードの判別にデータリンク層の識別子である MAC アドレスが使用できることも条件となる。また、大規模なネットワークを構築するために、1 回の遅延時間や帯域幅に対する変更処理がノード数によって増加しないことが重要となる。

### 4.5.2 Meteor のリンクエミュレータ

無線空間では、遅延時間や帯域幅は刻一刻と変化するため、短い時間で遅延時間や帯域幅を変更し続けることが求められる。ユーザ空間におけるリンクエミュレータの実装は、メモリコピーによるオーバーヘッドが精度に大きく影響すると考え、カーネル空間における実装を使用する。カーネル空間で動作可能なリンクエミュレータには TC/Netem と Dummynet が存在するが、Dummynet はネットワーク層で動作するリンクエミュレータである。そのため、イーサネットフレームに対する遅延挿入や帯域制限ができない。本論文で提案している Meteor はデータリンク層で動作することが重要であるため、Meteor

のリンクエミュレータとして利用できない。TC/Netem は、各ネットワークインターフェースの送信キューを利用するため、データリンク層で動作しイーサネットフレームに対する制御が可能である。また、Dummynet は遅延時間にミリ秒単位での指定であるのに対し、TC/Netem はマイクロ秒単位の指定が可能である。以上より Meteor においてはリンクエミュレータとして TC/Netem を使用する。

### 4.5.3 エミュレーション時のトラフィック制御

ユーザは、ネットワークエミュレータを使用する環境を自由に構築できるべきである。本節では、無線空間を作り出す上で考慮しつつ Meteor の動作時のトラフィック制御方法について検討する。

ネットワーク上での通信は、ユニキャストやブロードキャスト、マルチキャストに分類される。ユニキャストは 1 対 1 の通信であるため、送信元アドレスと送信先アドレスを指定することでフィルタリングの適用が可能である。しかし、ブロードキャストやマルチキャストのような 1 対多の通信では、送信先アドレスによるノードの識別が不可能となる。TC/Netem は送信時にフィルタリングを行うため、1 対多の通信はネットワークエミュレータ上を通過するフレームの向きを考慮する必要がある。

図 4.6 は、自ノード以外の送信元からの通信を全てフィルタリングし、他のノードからのトラフィックに対し遅延時間や帯域制限を行った際の処理を示している。図中の線は、NIC でイーサネットフレームを受信してからアプリケーションソフトウェアが受け取るまでの処理の流れとアプリケーションソフトウェアから送信されたイーサネットフレームが NIC から送信されるまでの処理の流れを示している。先に述べたように、ノードから送信するフレームに対して遅延挿入や帯域制限を行うとブロードキャストトラフィックをフィルタリングできない。そのため、受信時に送信元 MAC アドレスでフィルタリングすることにより、ブロードキャストに対応可能とする。

TC/Netem は、4.4.1 節で述べたように送信フレームにのみトラフィック制御が可能であるため、全ての受信フレームを一度 IFB インタフェースへ転送する。そして、IFB イ

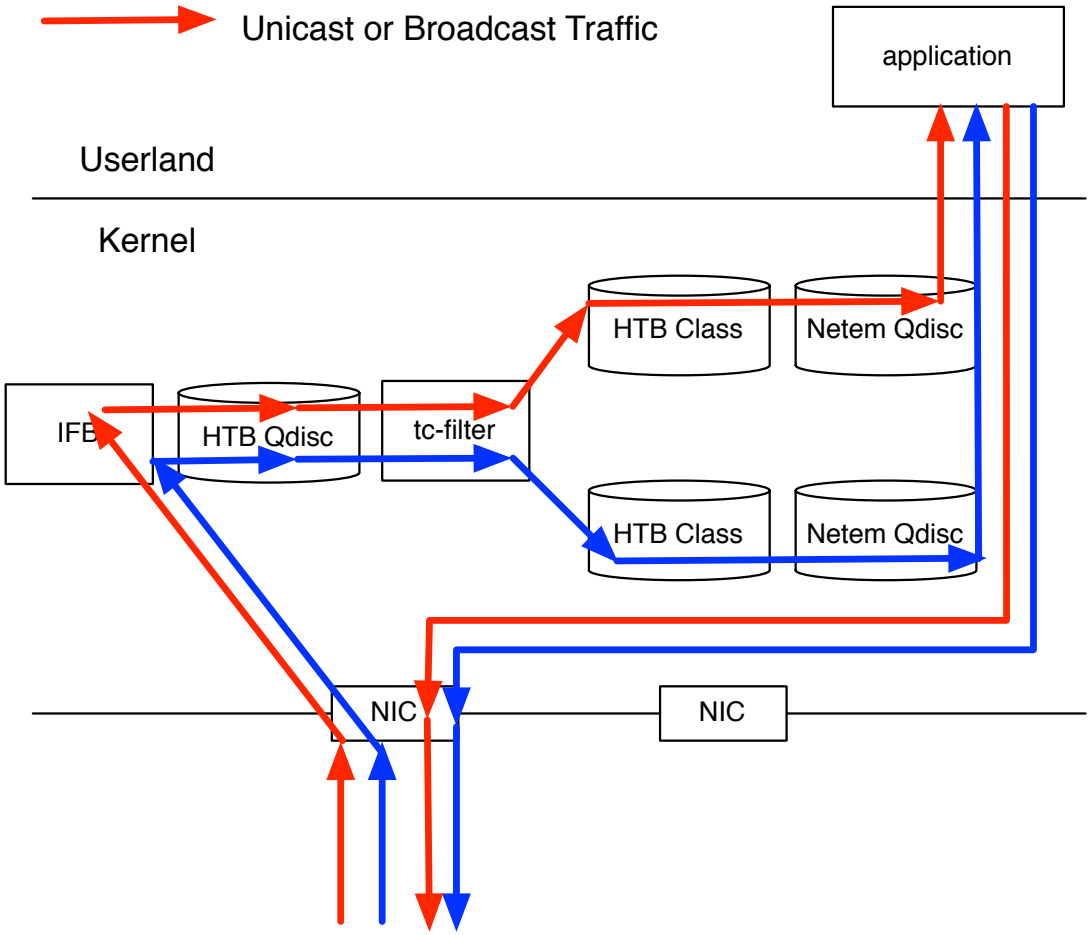



図 4.5: ノード上で動作時のトラフィック制御

ンタフェースからイーサネットフレームを再送信することで、受信フレームに対してトラフィック制御を行う。IFB インタフェースから再送信されるイーサネットフレームは送信処理時に HTB Class への振り分けのため tc-filter を通過する。フィルタリングの識別子には MAC アドレスや IP アドレス、任意のデータ識別子が指定可能であり、tc-ilter にマッチしたフレームは対応する HTB Class へと転送される。HTB Class では、帯域制限が行われ、デキューしたフレームは次に Netem Qdisc へと転送される。Netem Qdisc では、エンキュー時にパケット破棄の判定が行われる。破棄されずに Qdisc にエンキューされたフレームは指定時刻までキュー内に滞留し、その後アプリケーションへ送信される。

ここまで、ユニキャスト、ブロードキャストフレームを送信するアプリケーションソフトウェアを対象にエミュレーション手法を述べてきた。しかし、無線空間を通信するノー

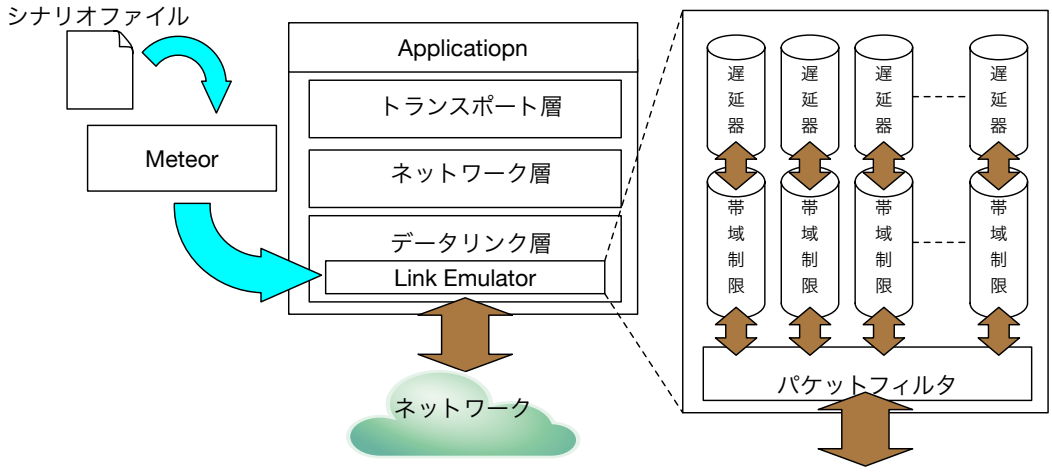



図 4.6: ノード上で動作時のトラフィック制御

ドが決まっているのであれば、全てのノード上で Meteor を動作する必要はない。Meteor を動作させる専用のノードを用意し、 1 対 1 のリンクを多数作り出すことでネットワークエミュレータの処理量を削減可能である。そこで Meteor では、ネットワークエミュレータを動作させるノードを専用に設置し、アプリケーションソフトウェアが動作しているノードからのトラフィックをブリッジする動作の実装も行った。

図 4.7 はブリッジでのトラフィック制御を行う際の動作を示している。受信したフレームは、ネットワークインタフェースより IFB インタフェースへ転送される。IFB は受信したフレームを自身の Qdisc(HTB Qdisc) へエンキューし、送信する時刻まで待つ。HTB Qdisc からデキューされたフレームは送信元アドレスと送信先アドレスでフィルタリングされ、次の HTB クラスへ転送される。HTB Class はノード間の帯域幅を制限する class モジュールであり、ここで帯域計算が行われる。Netem Qdisc では、エンキュー時に確率的にフレームが破棄され、破棄されなかったフレームのみエンキューされる。フレームの送信時刻は Netem へのエンキュー時に埋め込まれる。Linux のスケジューラにより、フレームが Qdisc からデキューされた後、埋め込まれた送信時刻を確認し、送信時刻に達していなければ Netem Qdisc へリキューし、送信時刻を過ぎていれば送信処理に移る。IFB インタフェースからのフレーム送信時に、自身の Forwarding DataBase(FDB) を確認し、送信する物理インタフェースの検索を行う。

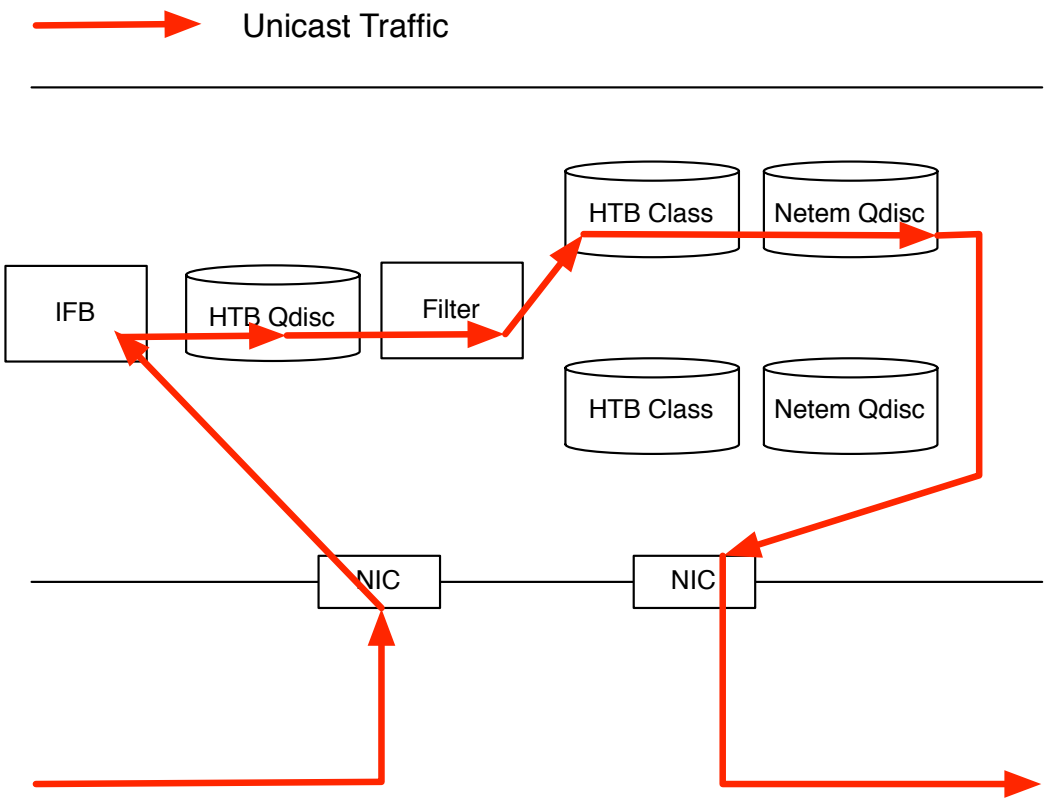



図 4.7: 中間ノードでのトラフィック制御

### 4.5.4 フィルタリング

ネットワークエミュレータが送信元ノードと送信先ノードの識別子として MAC アドレスや IP アドレスが挙げられる。Meteor には、データリンク層で動作する必要があることから MAC アドレスでのフィルタリング機能を実装している。しかし、MAC アドレスのみのフィルタリングでは IP アドレスでのフィルタリングで十分な検証でもノード毎の MAC アドレスを調べる必要がある。また、ハードウェアが変更した際に識別子を指定し直す必要が出てしまい、ネットワークエミュレータとして環境の可搬性が損なわれてしまう。そのため、Meteor の実装は IP アドレスによるフィルタリング機能も必要であると考え、Meteor ではフィルタリングの識別子に MAC アドレスと IP アドレスのどちらでも指定可能とした。

MAC アドレスや IP アドレスでのフィルタリングは、汎用ノード上でアプリケーショ

ンソフトウェアが動作することが前提となっている。しかし、組み込み機器などを動作させるために、Cooja/MSPSim [15] や mac80211_hwsim [12] などを使用する場合が考えられる。これらの技術は、汎用ノード上で組み込み機器を動作させ、無線フレームを送受信可能とする。無線フレームは有線ネットワーク上に送信できないため、ノードから有線ネットワークへ送信する際に、トランスポート層でカプセル化を行う。MAC アドレスや IP アドレスでフィルタリングを行うと無線フレームを送信したインタフェースの情報はカプセル化により隠蔽されてしまうため、送信元、送信先ノードを判断することができない。このようなカプセル化を行う通信では、カプセル化の形式やデータのペイロードは実装依存となるため、フィルタリング機能としてユーザが自由に識別子を指定できる必要がある。Meteor では、カプセル化された通信に対応するため、データリンク層のヘッダ情報からのオフセット値と 16 進数の値を指定することで、MAC アドレスや IP アドレスの他に自由な形でフィルタリングルールを記述可能とした。

### 4.5.5 タイマ制御

タイマ制御には、タイムスタンプカウンタ (TSC)、sleep 関数、イベンド駆動型などの方法が考えられる。sleep 関数やイベント駆動型は、変更時刻までの待機時間に CPU は使われない。sleep 関数は、カーネルの動的タイマとスケジューラを使用して実装されているため、ハードウェアタイマが Programable Interval Timer(PIT) の場合はソフトウェアのタイマ割り込み周期よりも短い時間での復帰はできない。例えば、タイマの割り込み周期が 250Hz であった場合、sleep からの復帰には 4ms 以上の時間が必要となる。これは、より高精度なハードウェアタイマである High Precision Event Timer(HPET) を使用することでよりマイクロ秒オーダでの割り込み可能である。しかし、HPET を使用するとハードウェア割り込みの回数が増加するため、CPU 負荷が高くなる。また、sleep 関数による待機方法は、指定時間プログラムの実行を遅延させるものであるため、シナリオに記述されている時間で sleep を実行するとエミュレータの動作時間が長くなってしまう。例えば、 Meteor の設定変更に必要な時間が 1ms、次の設定変更時間が 10ms 後の場

合、1ms の設定変更が行われた後に 10ms 待機するため、実際には 11ms 後に次の設定変更が行われる。sleep 関数によるタイマ制御を行うためには、Meteor 自身の処理時間を計測した上で待機時間を計算する必要がある。

TSC によるタイマ制御は、CPU のクロックカウンタを調べながら指定したクロックカウンタまで待機する方法である。TSC の読み出しには RDTSC と呼ばれる CPU 命令を使用する。TSC によるタイマ制御は CPU のクロックカウンタをベースに動作するため最も精度が高い。また、設定変更時間をクロックカウンタの値で指定するため Meteor の処理時間に関係なく動作することができる。一方で、SpeedStep や TurboBoost のような CPU 周波数を動的に変更する機能が有効になっているとクロックカウンタの増加値が変化するため正確な時間管理ができなくなる。そのため、TSC によるタイマ制御を行う際には、SpeedStep や TurboBoost を無効にしておく必要がある。

HPET を用いた sleep 処理で必要となる Meteor 自身の処理時間は gettimeofday で計測可能であるが、gettimeofday はシステムコールで重い部類の処理となる。そして、Meteor の処理時間は設定を変更する度に計測する必要があるため、計測処理のみでシステムコールの呼び出し回数が増大してしまう。本研究では、gettimeofday による負荷が大きいと考え、待機時間の指定が容易な TSC によるタイマ制御を使用することとした。

### 4.5.6 Meteor の実装

これまでの設計に基づき Meteor の実装を行った。Meteor は、 C 言語で実装されておりタイマ制御を行うライブラリである libtimer とシナリオパーサ、トラフィック制御ライブラリである libtc から構成される。libtimer は、TSC を用いて、シナリオに記述されているイベント時刻まで待機するライブラリである。シナリオパーサは、XML で記述された QOMET のシナリオファイルとの互換性と持たせるため、XML パーサライブラリである libexpat を用いている。既存の TC/Netem を利用したトラフィック制御は、libnetlink や libnl [38] を用いることでユーザランドから制御可能であるが、libnetlink は netlink socket を扱うのみであり、Qdisc を制御するためのプログラムは自身で実装す

る必要がある。一方で、libnl はトラフィック制御に関して未実装な部分があり、Meteor を実装するにあたり不十分であった。そこで本研究では、Meteor の実装にあたりトラフィック制御用ライブラリとして libtc の実装を行った。libtc は C 言語で実装されており、netlink socket を使用するため libnetlink を使用している。Ubuntu 12.04、14.04、Gentoo Linux(Kernel 2.6.32) にて動作確認を行った。なお、libtc や libtimer などを含めた Meteor は github にて公開している [39]。

## 4.6 Meteor の性能測定

### 4.6.1 機能面の評価

Meteor の評価として、Meteor、QOMET、MobiNet を比較し、機能面での評価を行った。評価項目として、ネットワークエミュレータが動作するレイヤや扱うことのできるノードの識別子とした。表 4.1 に機能評価の結果を示す。

ネットワークエミュレータが扱うことのできるレイヤは、Meteor がデータリンク層であるのに対し、QOMET はネットワーク層で動作する。MobiNet は遅延挿入や帯域制限はネットワーク層で行うが、ノードの間で MAC レイヤのエミュレーションを行うため、動作するレイヤをデータリンク層とネットワーク層の 2 つとした。しかし、MobiNet の MAC エミュレーションは無線チャネルを模倣する機構であり、ノード間の遅延や帯域幅はネットワークそうで行っているため、イーサネットフレームや IPv6 パケットに対する遅延挿入や帯域制限はできない。ネットワークエミュレータが扱うノードの識別子として、Meteor は MAC アドレスと IP アドレスをノードの識別子として用いることができる。MAC アドレスを識別子としてイーサネットフレームに対して遅延挿入や帯域制限を行うことにより、Meteor は IP ヘッダを持たないイーサネットフレームや IPv6 パケットに対応可能としている。対して、QOMET と MobiNet では IP アドレスのみを識別子として用いている。そのため、イーサネットフレームや IPv6 パケットに対する遅延挿入や帯域制限を行う機能は持っていない。

表 4.1: ネットワークエミュレータの機能評価

| | Meteor | QOMET | MobiNet |
|---|---|---|---|
| プロトコルレイヤ | データリンク層 | ネットワーク層 | データリンク層<br>ネットワーク層 |
| 識別子 | MAC アドレス<br>IPv4 アドレス | MAC アドレス<br>IPv4 アドレス | IPv4 アドレス |

### 4.6.2 設定変更の処理時間

本節では、Meteor がパラメータの変更を行うのに必要な処理時間について評価を行う。Meteor は、大規模な実験ネットワークで動作可能なネットワークエミュレータとして実装を行った。大規模な実験ネットワークで利用可能とするためには、ネットワークエミュレータが遅延時間や帯域制限の設定を変更する際の処理時間が重要となる。設定変更の処理時間が長いと無線空間を忠実に再現することが難しくなる。図 4.8 に 10 ノードから 5000 ノードまで 100 ノード単位で存在するノード数を増加させた無線環境をエミュレーションした際の設定に必要な時間を示す。1 台のノードが設定する数は自ノードを省いた N - 1 である。縦軸がパラメータ変更開始から終了までの処理時間、横軸がリンク数である。計測には gettimeofday を使用し、設定前と設定後の時刻差分を算出している。Meteor を動作させるために使用したノードは北陸 StarBED 技術センターのグループ I を使用した。グループ I の構成は表 4.2 の通りである。エミュレータノードは、OS に Ubuntu 14.04 をインストールし 2 つのインタフェース間で Meteor を動作させた。

Meteor が一度 TC/Netem へ設定するまでに必要な時間が約 17us 程度であった。これは、1 対 1 のノード間での通信を設定するために必要な時間である。複数の対向ノードを扱いたい場合、Meteor は TC/Netem へ連続して制御を行うため、ノード数の増加に対して処理時間は線形に増加する。そのため、1 台の Meteor が 4999 台のノードへの接続をエミュレーションするにはパラメータ変更間隔は約 90ms 必要であることがわか

る。これに対し、QOMET は 1500 台以下のノードへの変更であれば Meteor と比較し短い時間で動作している。しかし、ノードの台数が増加すると徐々に処理時間が増加していき、1500 台以降は Meteor より処理時間が長くなっている。これは、Meteor で使用している TC/Netem が構成情報を木構造で管理しているのに対し、QOMET で使用している Dummynet はリストで管理しているためであると考えられる。

TC/Netem と Dummynet を比較した場合、カーネルモジュールに送信するまでの処理は TC/Netem のほうが複雑である。そのため、 1 台のノードへの設定時間のみを計測すると TC/Netem を利用する Meteor の負荷が高くなる。今回使用したノードでは、1500 台までの環境で QOMET のほうが高速に動作している。しかし、Dummynet が構成情報をリストで管理しているため、扱うノードの数が増加するとカーネルモジュールの負荷が高くなる。結果、Meteor ではノードの数に対して線形に増加しているのに対して、QOMET では徐々に処理時間が増加している。そのため、1500 台以下であれば QOMET が高速に動作するが、1500 台以上であれば Meteor が有利となる。

### 4.6.3 エミュレーション精度

#### 単純な波形での模倣

本論文で設計・実装した Meteor の精度を計測するため評価実験を行った。評価環境は、図 4.9 に示すように、トラフィックジェネレータである Spirent Communications SmartBits 600B とエミュレータノードを 2 本の UTP ケーブルで接続した。実験には、IPv4 ヘッダを含めた状態で SmartBits 600B が送信可能な最小サイズである 70Bytes とし、1 秒間の送信フレーム数は 802.11g の最大帯域となる 54Mbps を送信した場合の 96kfps とした。送信フレーム数やフレームのサイズを変更した実験も行ったが、結果に差異が見られなかったため、本検証結果には最小フレームサイズで最大送信数となる 96kfps の結果のみとした。SmartBits 600B から送信されたイーサネットフレームはエミュレータノードで遅延挿入、帯域制限が行われ SmartBits 600B に折り返す構成となる。SmartBits 600B から送信されるイーサネットフレームには、送信時刻が埋め込まれ

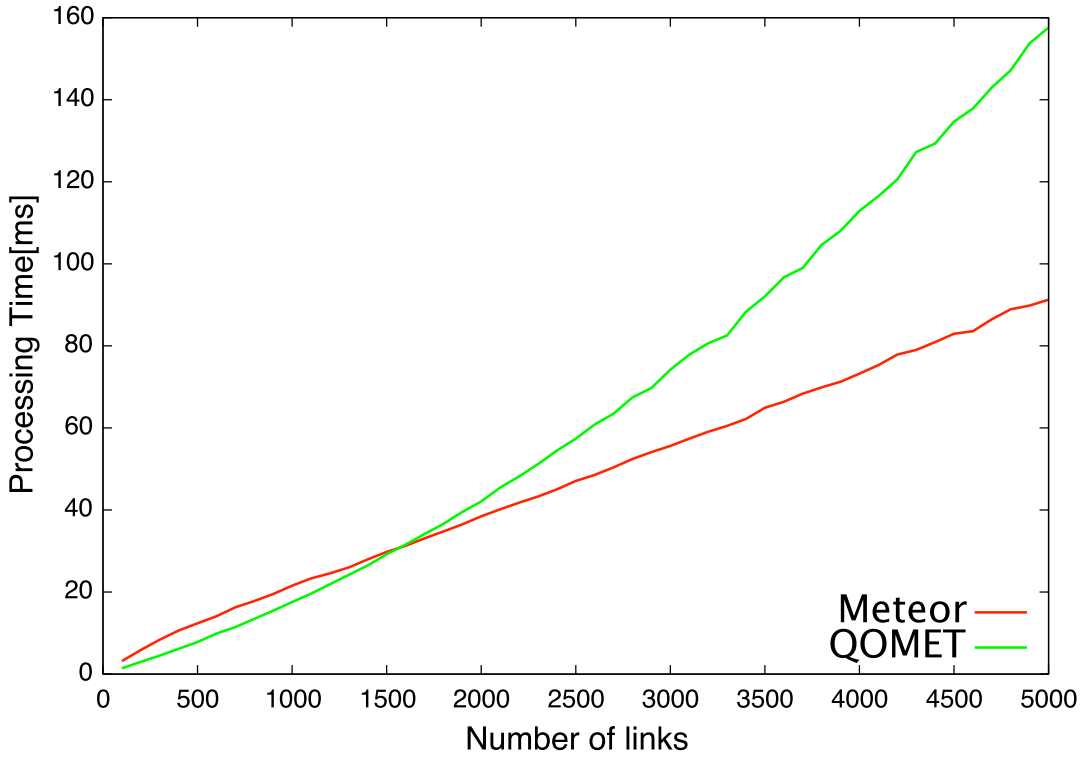



図 4.8: Meteor の処理時間

表 4.2: 実験ノードのスペック

| Model | Cisco UCS C200 M2 |
|---|---|
| Chipset | Intel 5520 (Tylersburg) chipset |
| CPU | Intel 6-Core Xeon X5670 x 2 |
| Memory | 48GB |
| NIC | Broadcom 5709 Quad Port |

ており、受信時に受信時刻との差分を出すことで遅延時間を算出した。

Meteor の設計から遅延時間の変動幅によってネットワークエミュレータとしての精度に影響があると考えられる。例えば、遅延時間の変動が小さければ変更時にフレーム送信処理の負荷も小さくなると考えられるため精度は高くなる。一方で遅延時間が大幅に短くなれば、キューに滞留しているフレームを一斉に送信する必要があり精度が低下すると考

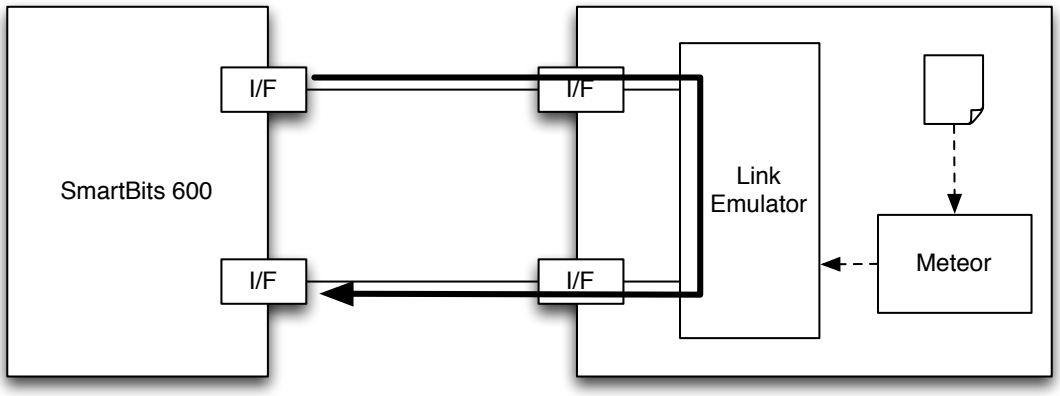



図 4.9: 検証環境

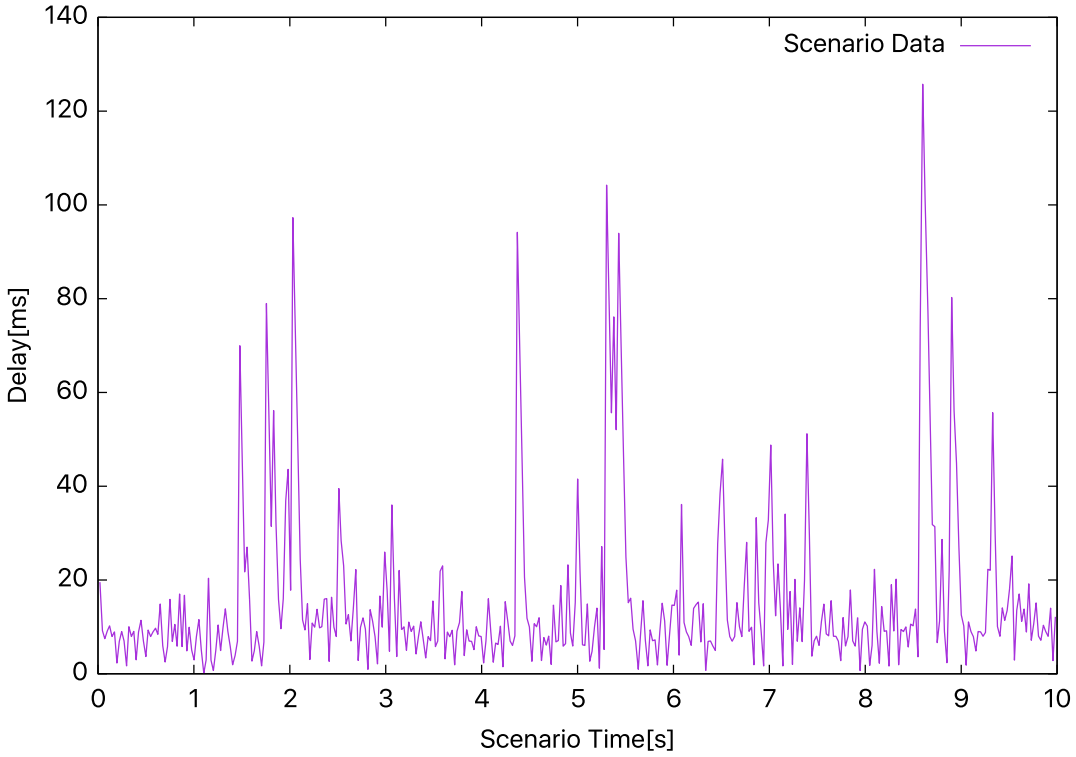



図 4.10: Meteor のシナリオデータ

えられる。シナリオは図 4.10 に示すように遅延時間が変動し続けるものを使用した。

図 4.10 を使用して実際に Meteor と QOMET を動作させ、SmartBits にて遅延時間の変化を計測した。計測では、無線ノードを 32 から 1024 台まで増加させた際の精度を計測した。

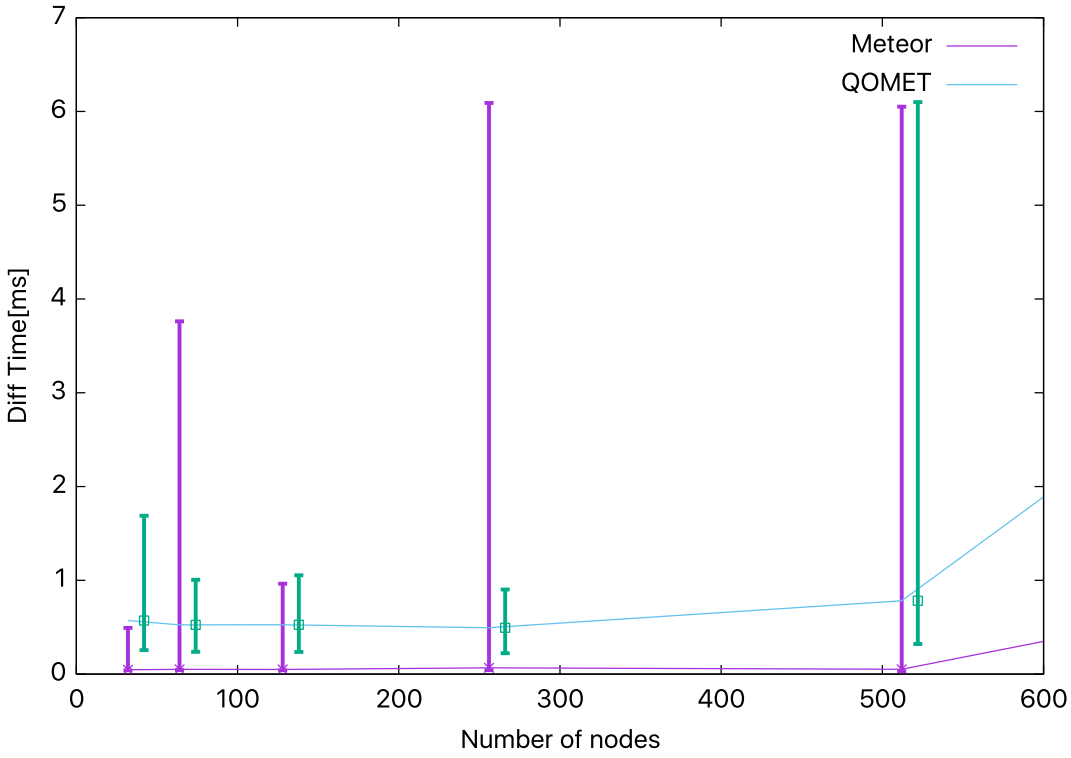



図 4.11: Meteor: ノード数とエミュレーション精度

Meteor の計測結果を図 4.11、QOMET の計測結果を図 4.12 に示す。計測結果は、シナリオデータとの遅延時間の差分を CDF にて示している。

Meteor は、ノード数の増加に対して徐々に遅延時間の誤差が大きくなっているのに対し、QOMET では、256 ノードまでは大きな変化がないものの、512 ノードで精度の低下が見られる。そして、1024 ノードではエミュレーション精度が大きく低下していることがわかる。これは、Meteor と QOMET が使用しているリンクエミュレータが各ノード毎の伝搬パラメータを保持する方式の違いによるものであると考えられる。また、Meteor、QOMET のどちらも 100 ms の遅延誤差が出ているが、これは前節で述べた処理遅延によるものと、長い遅延時間から短い遅延時間に伝搬パラメータを変更した際の挙動によるものであると考えられる。

また、ノード数が少ない場合でも、QOMET は遅延時間の誤差が大きく出ている。図 4.13 に、ノード数が 512 までの中央値、第一四分位点、第三四分位点を示す。QOMET は、第一四分位点、第三四分位点の値に大きな変化がないものの、中央値の値が大きい。

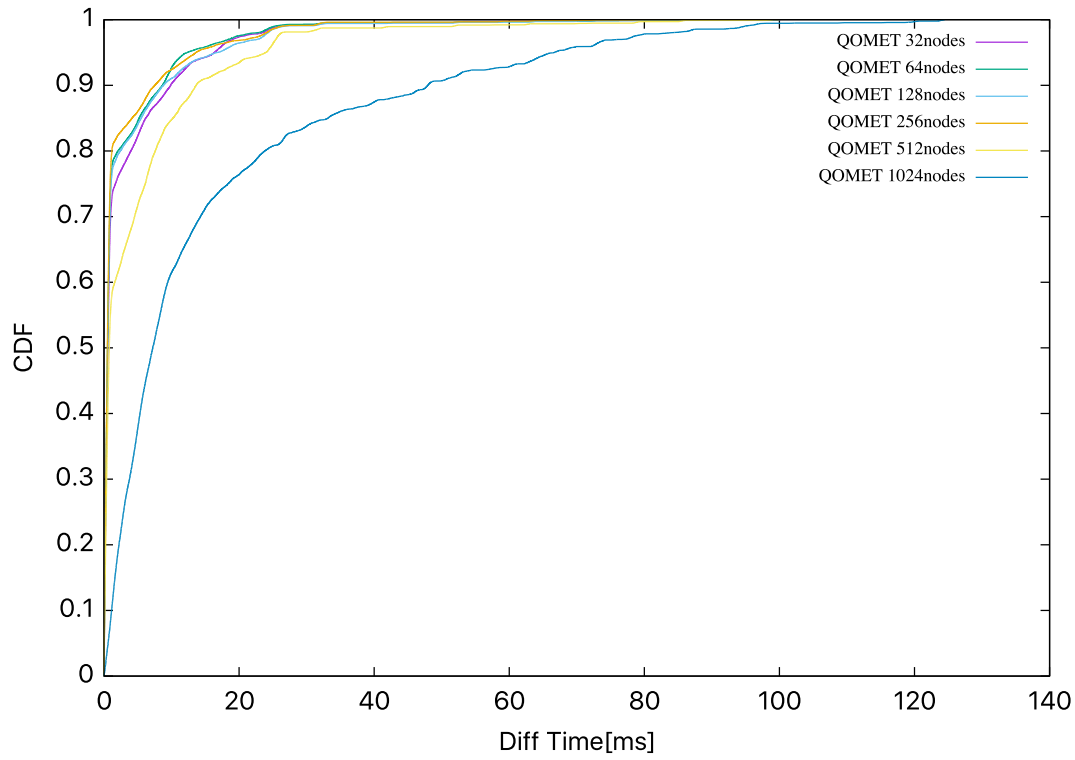



図 4.12: QOMET: ノード数とエミュレーション精度

これに対し、Meoteor では中央値の値が小さいが、四分位点の誤差が大きい。

QOMET が使用している ipfw は、伝搬パラメータを変更する際に、エミュレーションルールを作成し直し、参照ポイントを切り替える動作となっている。そのため、ノード数が少ない場合でも遅延誤差が大きく出ているものと考えられる。一方で、Meteor は、木構造で構成された 1 つの伝搬パラメータを変更するのみでよいため、遅延設定による全体に対する影響は少ない。しかしながら、1 つの伝搬パラメータを変更する際に、ロック制御が入ってしまうため、ある伝搬パラメータを変更している間、その伝搬エミュレーションは停止してしまう。そのため、タイミング次第では遅延時間が大きく延びてしまっている。このロック制御による遅延時間は、現状避けがたい問題である。しかし、Linux では lockless qdisc の実装が行われており、この実装を用いれば遅延時間の誤差を小さくすることが可能であると考えられる。

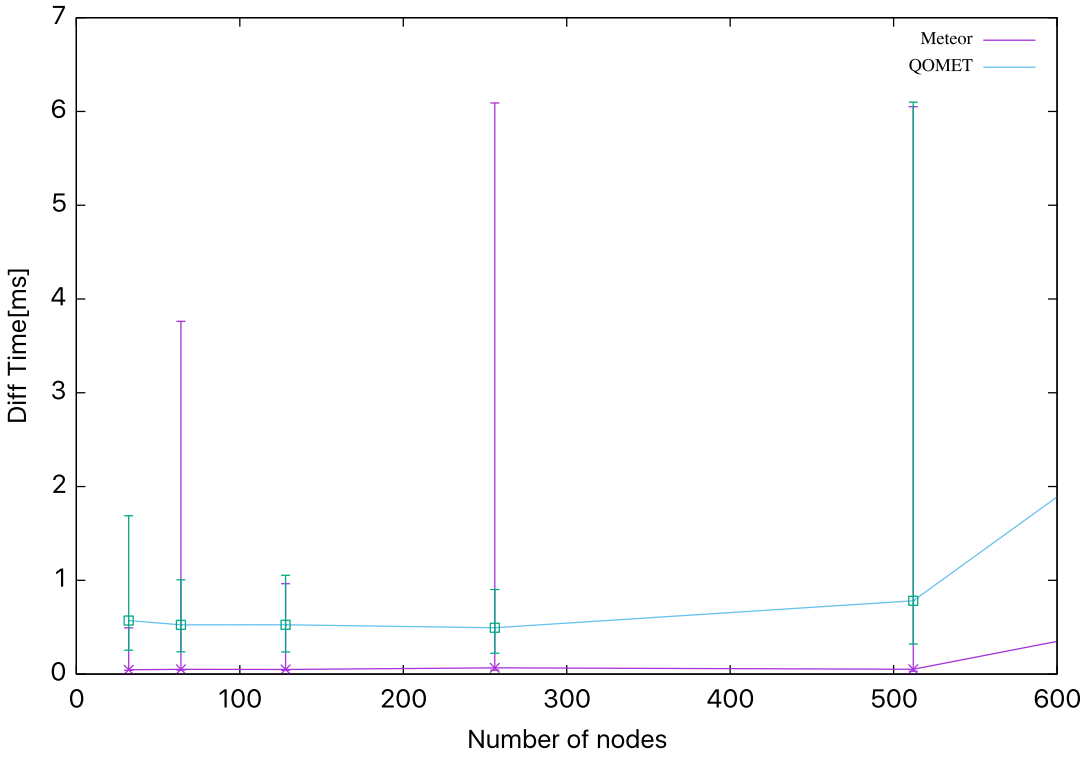



図 4.13: 中央値、第一四分位点、第三四分位点の比較

### Meteor を用いたメッシュネットワークの構築

本節では、Meteor を用いた構築したネットワーク上で 802.11s、 AODV、B.A.T.M.A.N、OLSR のルーティングプロトコルを動作させ、アドホックネットワーク上でのマルチホップ時に遅延時間の変化の計測を行った。図 4.14 は、10 台のノードを用いて構築したメッシュネットワークである。

802.11s には、open80211s [40] を使用した。open80211s を使用するためには無線インタフェースが必要となるため、仮想インタフェースを作成できる mac80211_hwsim も用いている。

表 4.3 は、各ルーティングプロトコル動作時のホップ数毎の遅延時間である。各ルーティングプロトコルでホップ数が増加する毎に遅延時間も増加していることがわかる。B.A.T.M.A.N のみ遅延時間の増加が大きくなっているのは、データの転送処理が他のルーティングプロトコルと比較して大きいためであると考えられる。OLSR(HNA6)

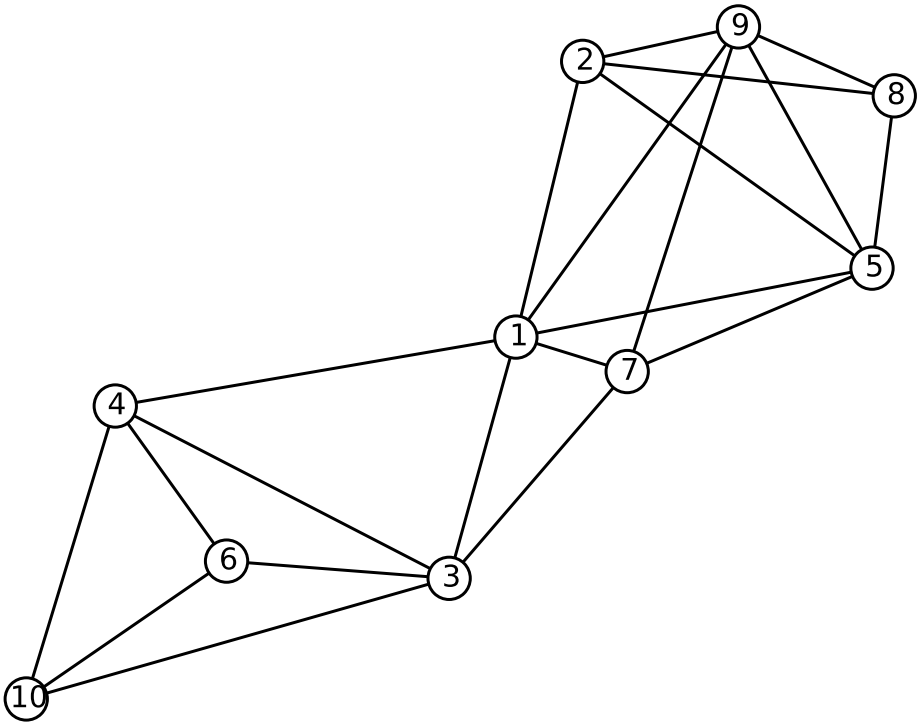



図 4.14: 10 台でのメッシュネットワーク

表 4.3: マルチホップ通信時の遅延時間 [ms]

| Routing Protocol | 1 Hop | 2 Hop | 3 Hop | 4 Hop |
|---|---|---|---|---|
| AODV | 32.1 | 97.4 | 140.2 | 153.7 |
| 802.11s | 32.3 | 98.4 | 142.0 | 158.0 |
| B.A.T.M.A.N-adv | 32.3 | 150.0 | 229.0 | 294.0 |
| OLSR(HNA6) | 32.4 | 98.0 | 141.1 | 154.9 |

は、IPv6 を用いて動作させた結果である。Meteor は、ネットワーク層のプロトコルに依存しないネットワークエミュレータとして実装を行ったため、IPv6 を用いたルーティングプロトコルである OLSR(HNA6) やレイヤ 2 ルーティングプロトコルである B.A.T.M.A.N-adv のような、IPv4 ヘッダを持たないフレームにも対応できている。

### 4.6.4 スケーラビリティ

本節では、仮想ノード 1024 台を用いて格子状のネットワークを構築し、Meteor の動作検証を行う。図 4.15 は、1024 台の仮想ノードを格子状に配置したネットワーク図である。各ノード間の距離は 30m とし、この時のノード間の遅延時間は 30ms となる。

検証には、OLSR(HNA6) を使用した。AODV は、RFC にて最大ホップ数が 35 までと決められている。そのため、1024 台のノードを格子状に配置した場合、最大ホップ数は 63 となり、AODV では到達できない規模のネットワークとなる。Meteor のみを動作させた時の CPU 使用率は 8 % であったが、OLSR を動作させると CPU 使用率が 100 % となり、パケットロスや想定より大きな遅延が発生した。OLSR の HELLO メッセージは標準の設定では 6 秒間に 1 回であるため、HELLO メッセージのみであればトラフィック負荷は高くない。しかし、OLSR を 1,000 台規模で動作させると大量の Topology Control(TC) メッセージが送信される。そして、TC メッセージを受信する度に経路表の更新が行われるため、各ノードの負荷が高くなっていると考えられる。

## 4.7 おわりに

インターネットへの接続端末が多様化する中、ネットワークテストベッド上に擬似的な無線空間を作り出すネットワークエミュレータが数多く提案されている。しかし、新たな無線技術の提案によりネットワーク層で動作しているネットワークエミュレータでは対応が難しくなってきている。

そこで我々は、無線空間における遅延時間や帯域制限を忠実に再現するとともに、アプリケーションソフトウェアのプロトコルに依存しない無線空間のエミュレーション手法の提案を行った。また、提案手法を実現するネットワークエミュレータである Meteor の設計・実装を行った。Meteor は、既存研究の QOMET や MobiNet と比較しネットワーク層での動作からデータリンク層での動作にしたため、これまで検証が困難であった IEEE802.11s や AODV のようなアプリケーションソフトウェアの検証が可能となった。

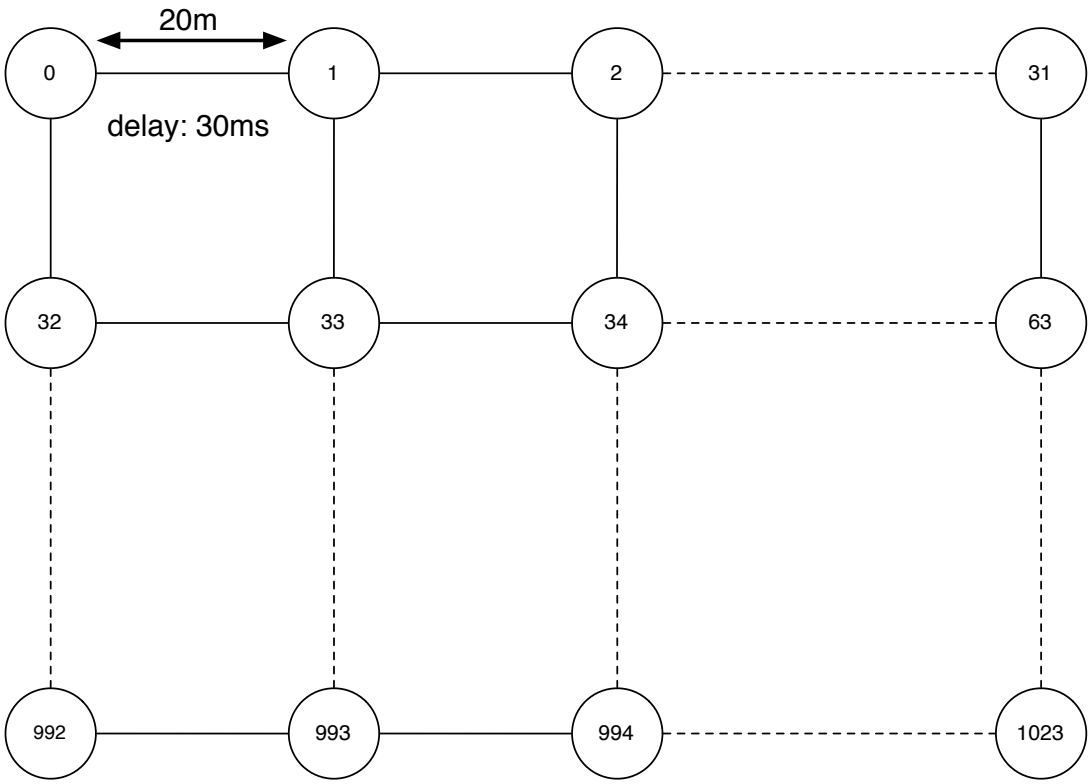



図 4.15: 1,024 台でのメッシュネットワーク

既存のネットワークエミュレータは、ノードの識別にネットワーク層の識別子である IP アドレスが用いられているが、Meteor ではユーザがパケットの中のデータを自由に指定できる形でアプリケーションの実装依存となる特殊なデータ形式においても対応可能としている。識別子としての指定方法は、16 進数の値で指定するため実験者が容易に設定できるとは言い難いが、Cooja/MSPSim や mac80211_hwsim のようなカプセル化されたパケットにおいてもエンドノードの識別が可能である。

Meteor を用いたネットワークエミュレータの精度は、遅延時間が大きく減少する際に追従性が低下する傾向が見られるものの、ハンドオーバが発生した際の遅延時間の増加に対しても再現できている結果が得られた。本提案を用いることにより、これまで困難であったデータリンク層のプロトコルを使用するアプリケーションソフトウェアの検証が可能な擬似的な無線空間を作り出すことが可能となる。

# 第 5 章

# Asteroid: 仮想無線ネットワークエミュレータ

## 5.1 はじめに

無線ネットワーク技術は、ノート PC や携帯電話、スマートフォン、IoT のような小型デバイスをインターネットへ接続するために広く使われている。小型デバイスは、利便性、高可用性の向上のため、様々な技術が導入されている。この技術の中には、これまでのイーサネットではなく、無線ネットワーク技術特有なものがある。Meteor や QOMET のような伝搬エミュレータは、無線空間での伝搬特性を再現するであり、通信にはイーサネットフレームが使用される。しかし、無線ネットワークで使用される技術には、認証やレイヤ 2 メッシュプロトコルのようなイーサネットフレームでは利用できないものがある。このような無線ネットワーク特有の技術は、伝搬エミュレータだけでは再現できない。

このような無線技術を扱うエミュレータとして、Mac80211_hwsim [12] や Cooja [15] などが提案されている。本研究では、これらをハードウェアエミュレータと呼ぶ。ハードウェアエミュレータは、1 台のサーバ上で動作し、無線フレームを用いた通信を可能とする。この無線通信は、無線ネットワークで使用される管理フレームを扱うため、RSSI や

SSID のような無線独自の情報を扱うことができる。しかし、無線通信を忠実に再現しているがゆえに有線ネットワークで接続された複数のサーバ間での通信が行えない。そのため、ハードウェアエミュレータが持つ規模追従性は、動作させるサーバの性能に依存し、最大でも 100 台が限界となる。

本章では、ハードウェアエミュレータが抱える規模追従性に関する問題を解決する仮想無線ネットワークエミュレータ Asteroid について述べる。Asteroid は、ハードウェアエミュレータが送受信する無線フレームをカプセル化し、有線ネットワーク上で送受信可能とする。また、無線フレームの送信に必要となる時間の制御を行うことで、擬似的な衝突の再現を行う。これにより、アプリケーションソフトウェアへの API の提供、無線通信時の振る舞いを再現しつつ有線ネットワーク上に仮想的な無線ネットワークの構築を実現した。

## 5.2 ハードウェアエミュレータの問題点

ハードウェアエミュレータは、サーバ上でモバイル端末や無線インタフェースの再現する。ハードウェアエミュレータの実装には、Mac80211_hwsim [12] や Cooja [15] などがある。本節では、ハードウェアエミュレータが持つ問題点について述べる。

mac80211_hwsim や Btvirt のような無線インタフェースを提供するハードウェアエミュレータは、Linux カーネルモジュール、ユーザランドソフトウェアとして提供されているが、これらが送受信するのは無線フレームであるため、有線ネットワークを介した通信ができない。そのため、単一のホスト上でしか動作せず、規模追従性がホストの性能に制限される。また、mac80211_hwsim を用いている Mininet-wifi は、mininet を同様の規模追従性があると述べられているが、多くの無線インタフェースを扱った場合、無線インタフェースの制御負荷が大きくなるため規模追従性は低下する。図 5.1 は、複数コンテナを起動させた状態の CPU 使用率、1 秒間のコンテキストスイッチ回数を示した図である。コンテナ数が 32 以降、コンテキストスイッチの回数が増加し始め、100 では CPU 使用率が 100% となっている。このことからハードウェアエミュレータを用いて単一ホ

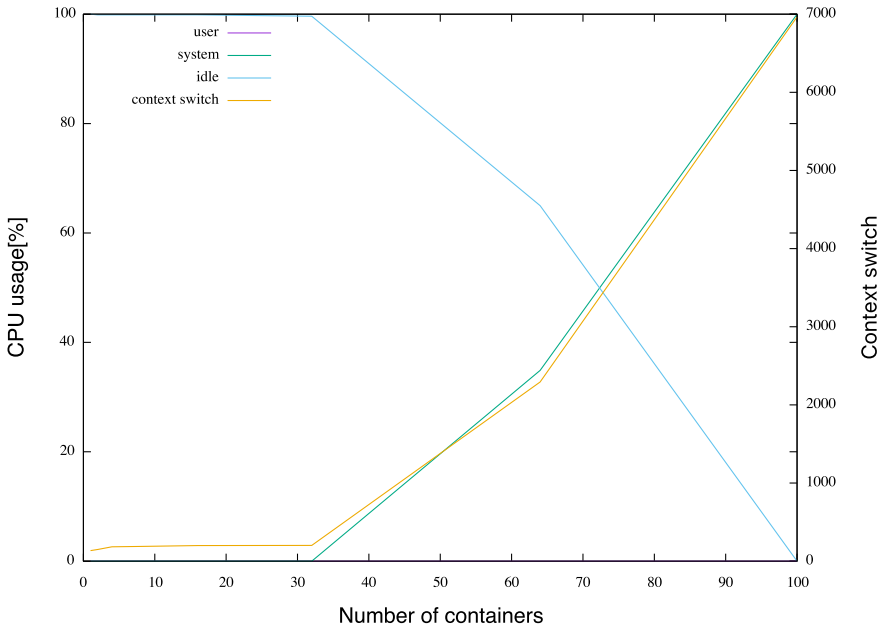



図 5.1: コンテナ数の増加による CPU 使用率の変化

スト上で多数の無線インタフェースを扱うとコンテキストスイッチの回数が増加し、リアルタイムでの動作が保証できなくなり、アプリケーションソフトウェアの動作へ影響が出るといった問題が発生する。

Cooja もエミュレーションされた端末が有線ネットワークで接続されている構成であれば、複数台の計算機を用いた環境が構築可能であるが、この場合は無線インタフェースを用いた通信は行えない。加えて、Cooja も多くの端末をエミュレーションした場合、CPU 負荷が高くなりリアルタイムでの動作が保証できなく、検証したいアプリケーションソフトウェアへの影響が出るといった問題が発生する。これは、Classic Bluetooth の最大コネクション数が 16 となっているためであると考えられるが、Bluetooth Low Energy(BLE) のような多数の無線ノードが情報を発信するような環境を構築しての検証はできない。

これらの問題は、複数の無線インタフェース、デバイスをエミュレーションした際に発生していることから、単一ホスト上ではなく、複数ホストで無線ネットワークを再現することで解決可能である。

## 5.3　有線ネットワーク上での無線通信

### 5.3.1　Asteroid

有線ネットワーク上で擬似的な無線空間を再現するためには、無線ノード間の遅延時間や帯域幅、パケットロス率の伝搬特性を再現する手法が多く採用されている。アプリケーションソフトウェアから見た場合、通信先との間で発生する伝搬特性を再現することで無線ネットワークを介して通信しているように見せかけることができる。本論文では、大規模かつネットワークプロトコル非依存な伝搬エミュレータとして Meteor の設計と実装と行った。しかし、新たな無線技術の中には有線ネットワークとは異なる技術で通信路の確保を行うものがある。

無線ネットワークは、基本的にブロードキャストでの通信を行うためセキュリティの観点から認証を使用しての接続が一般的に用いられている。無線ネットワークの認証技術は、プロトコル仕様に含まれている。また、無線ノードのみで構成する無線メッシュネットワーク技術には、無線フレームによるメッセージ交換で経路を構築するものがある。このような技術は有線ネットワークでは使用できないプロトコルであるため、伝搬エミュレータのみを使った擬似的な無線空間の再現では扱うことができない。そのため、仕様で定義されたプロトコルを用いた技術をネットワークテストベッドで使用するためには、無線フレームによる通信路を有線ネットワーク上に構築する必要がある。

asteroid は、無線空間での衝突を再現しつつ、有線ネットワーク上で無線フレームの送受信を可能とする仮想無線ネットワークエミュレータである。図 5.2 は、Asteroid を用いて複数の汎用計算機間が有線ネットワーク上で無線通信を行っている。node A と Node B は AccessPoint を介して通信可能な状態である。

#### 衝突の再現

実世界の無線ネットワークでは、無線ノードから同時に無線フレームを送信した場合、電波の衝突が発生する。実際の無線ネットワークでは、衝突の検知は不可能であるためプ

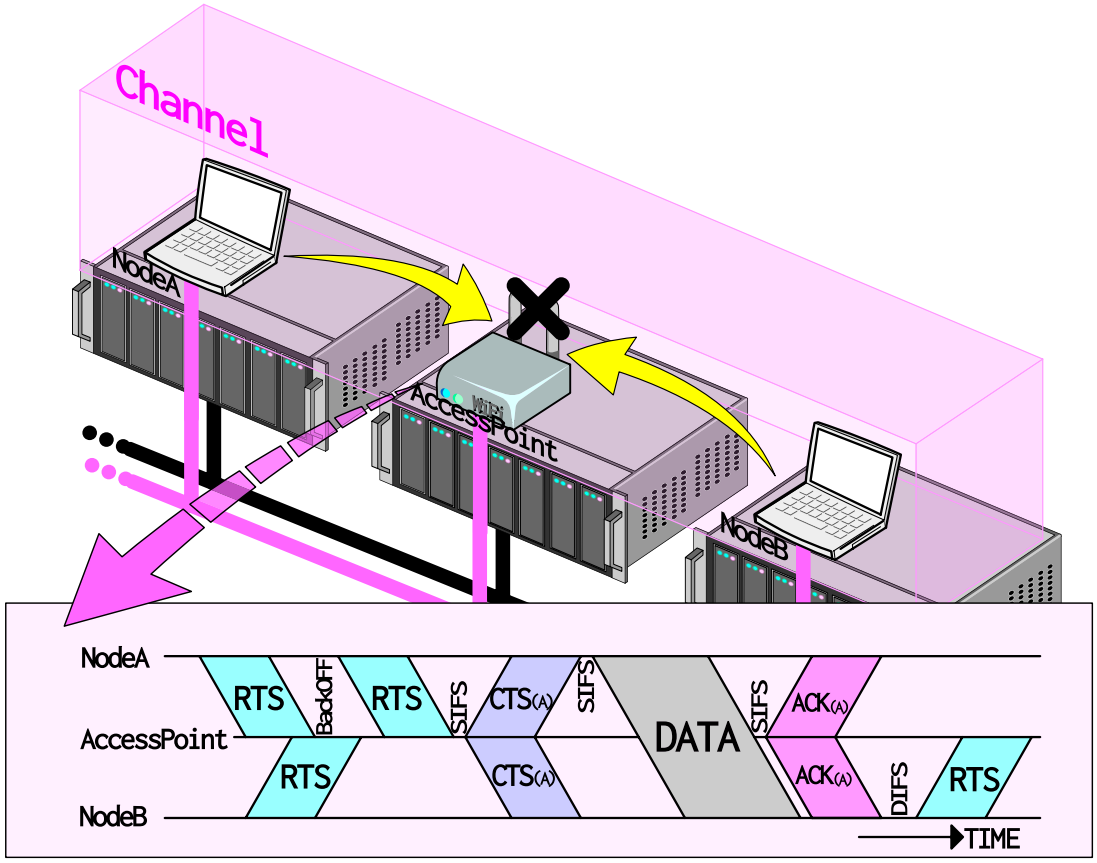



図 5.2: Asteroid のデザイン

ロトコルによって決められた時間相手からの応答がなかった場合、BackOFF 時間待機した後データの再送を行う。

無線ネットワークでは、衝突回避の方法に CSMA/CA と CSMA/CA with RTS/CTS が存在するが、CSMA/CA は隠れ端末問題に対応できないため、実際の無線ネットワークでは CSMA/CA with RTS/CTS が用いられることがほとんどである。本論文では、CSMA/CA with RTS/CTS を RTS/CTS と呼ぶ。rTS/CTS は以下の手順で衝突を回避する。

1. ノード A は データを送信する前に衝突が発生しないよう RTS フレームをブロードキャストする。
2. RTS フレームを受信したアクセスポイントは、CTS フレームをブロードキャストすることでノード A がデータを送信することを他のノードへ通知する。

3. CTS フレームを受信したノード A は、データを送信する。他のノードはノード A のデータ送信が完了するまで待機する。
4. ノード A のデータ送信が完了した後、アクセスポイントは ACK フレームをブロードキャストする。この ACK フレームによって他のノードはノード A のデータ送信が完了したと判断する。

rTS/CTS は、予めデータを送信する権利を得ることでデータ送信中の衝突の回避しているが、RTS フレームの衝突は回避できない。

図 5.2 は、Node A と Node B は AccessPoint と通信している際に、Asteroid が電波の衝突を再現している様子を示している。node A と Node B は、AccessPoint へ RTS フレームを送信しているが、電波の衝突が発生したため AccessPoint は正しいデータを受信できない。

asteroid では、この衝突を再現するため無線フレームの送信中に他の無線ノードから受信した場合、受信フレームを破棄する。

### 5.3.2 データ構造

無線フレームを他ノード間で交換するためには、一度無線フレームを有線ネットワーク上で交換可能なフォーマットへ変換する必要がある。この方法には、トランスレーションとカプセル化の方法が考えられる。トランスレーションを行うためには、無線フレームから IEEE 802.11 ヘッダの情報を読み取り、何らかの形に変換する必要がある。これが IEEE 802.11 のみの対応であればトランスレーションによる手法で問題ないが、実環境で使用される無線技術は Bluetooth や Zigbee なども存在する。これら数種類の通信メディアのヘッダ情報を読み取り適切なフォーマットへの変換をフレーム毎に行うことは困難である。また、無線技術で使用されるヘッダはイーサネットと比較して多くの情報が格納されるため、イーサネットヘッダ以外の場所に何らかの形で情報を格納する必要がある。これに対し、カプセル化は無線フレームをそのまま扱えるため処理は簡素になる。また、カプセル化用のデータコンテナにはメディアタイプのフィールドを付与させることで

通信メディアの識別も可能となる。

asteroid は無線フレームをカプセル化ることにより、ハードウェアエミュレータから送信された無線フレームを有線ネットワークを介して透過的に送信することができる。asteroid はハードウェアエミュレータの通信を有線ネットワーク上で実現するため Wi-Fi Protected Access 2(WPA2) やホットスポット、IEEE 802.11s などの従来の無線伝搬エミュレータでは処理できない無線ネットワーク技術を使用することが可能となる。

## 5.4 フレーム送信時間のエミュレーション

IEEE 802.11 では、11b や 11a/g で異なる変調方式が使われる。例えば、IEEE 802.11g では、1500Bytes のフレームを送信するのに 252$\mu$s 必要である。asteroid は、このフレーム送信時間を再現することで、無線ネットワークのリンク速度の再現と衝突の再現を行う。

IEEE 802.11b では、DSSS が使用され、11a/g では OFDM が使用される。

無線フレームを送信するのに必要な時間は、以下の計算式で求めることが可能である [41] 。

$$CW = CW_{min} T_{slot} \tag{5.1}$$

データフレームのフレーム送信時間 (DSSS)

$$T_{D\_DATA} = T_P + T_{PHY} + \frac{8L_{H\_DATA} + 8L_{DATA}}{100000 R_{DATA}} \tag{5.2}$$

データフレーム確認応答のフレーム送信時間 (DSSS)

$$T_{D\_DATA} = T_P + T_{PHY} + \frac{8L_{ACK}}{100000 R_{ACK}} \tag{5.3}$$

データフレームのフレーム送信時間 (OFDM)

$$T_{D\_DATA} = T_P + T_{PHY} + \tag{5.4}$$

$$T_{SYM} \frac{16 + 6 + 8L_{H\_DATA} + 8L_{DATA}}{N_{DBPS}} \tag{5.5}$$

表 5.1: フレーム送信時間の計算に必要なパラメータ

| パラメータ | 意味 |
|---|---|
| CW(Contansion Window) | 送信可能までも待ち時間 |
| $CW_{min}$ | CW の最小値 |
| $T_{slot}$ | スロットタイム |
| $T_P$ | Physical preamble の送信に必要な時間 |
| $T_{PHY}$ | PHY header の送信に必要な時間 |
| $H_{DATA}$ | MAC 副層のデータ長 |
| $L_{DATA}$ | ペイロードのデータ長 |
| $R_{DATA}$ | 通信レート |
| $N_{DBPS}$ | OFDM のシンボル長 |

表 5.2: 固定パラメータの値

| パラメータ | 802.11a | 802.11b |
|---|---|---|
| $T_{slot}$ | $9\mu$ | $20\mu$ |
| $T_P$ | $16\mu$ | $144\mu$ |
| $T_{PHY}$ | $4\mu$ | $48\mu$ |
| $CW_{min}$ | 15 | 31 |

データフレーム確認応答のフレーム送信時間 (OFDM)

$$T_{D\_ACK} = T_P + T_{PHY} + T_{SYM}\frac{16 + 6 + 8L_{ACK}}{N_{DBPS}} \tag{5.6}$$

### 5.4.1　アプリケーションソフトウェアへの API の提供

無線ネットワークを使用するアプリケーションソフトウェアには、オペレーティングシステムが提供する標準的な API を用いるものがあるため、Asteroid による仮想無線ネッ

トワークを再現した上でもこの API を提供する必要がある。asteroid が提供する仮想無線ネットワークは、ハードウェアエミュレータを用いているためオペレーティングシステムへ提供可能な API はハードウェアエミュレータと同じとなる。

## 5.5 Asteroid の実装

本節では、Asteroid の実装について解説する。asteroid は、IEEE 802.11 を扱うハードウェアエミュレータである mac80211_hwsim と連携して動作する。

### 5.5.1 mac80211_hwsim との連携

mac80211_hwsim は、カーネルモジュールと生成された無線インタフェース間で通信を行う。そのため、カーネルモジュールの読み込みだけでは無線フレームを取得することはできない。mac80211_hwsim には、wmediumd モードと呼ばれるユーザ空間へ無線フレームを転送する仕組みが実装されている。wmediumd は、mac80211_hwsim 用に実装された無線空間中の伝搬品質を再現するためのソフトウェアである。asteroid は、この wmediumd モードを使用して無線インタフェースから送信された無線フレームを取得する。

### 5.5.2 カプセル化フォーマット

無線フレームを他ノード間で交換するためには、一度無線フレームを有線ネットワーク上で交換可能なフォーマットへ変換する必要がある。この方法には、トランスレーションとカプセル化の方法が考えられる。本研究では、多数のフレームフォーマットへの対応が容易なカプセル化を採用した。また、無線空間ではブロードキャストが多く使用されることから、Point-to-Point でコネクションを必要とする TCP では管理が難しくなる。そのため、UDP によるカプセル化が望ましいと考えた。

図 5.3 は、Asteroid がカプセルする際のフォーマットである。l2TP や Virtual eXtensible Local Area Network(VXLAN) [42] のように UDP によるカプセル化を行うトン

ネリング技術は複数提案されている。これらのトンネリング技術は、イーサネットフレームを対象としており、セッション ID やセグメント ID による識別ができればオリジナルのイーサネットフレームをカプセル化すればよい。ただし、これはイーサネットフレームをカプセル化する場合であり、ハードウェアエミュレータが送受信する無線フレームのカプセル化するためにはいくつか考慮しなければならない点がある。例えば、IEEE 802.11 の無線フレームから送信元アドレスや送信先アドレス、シーケンス番号のような情報は IEEE 802.11 ヘッダから読み取ることができるが、ハードウェアエミュレータの実装依存となるパラメータは無線フレームから読み取ることはできない。asteroid ではこのような実装依存の情報を扱えるようメタコンテナが必要であると考えた。本論文では、UDP でのカプセル化および、Meta Container の実装から Geneve 形式のフォーマットでカプセル化を行った。geneve は、IETF の Network Virtualization Overlays(nvo3) ワーキンググループで議論されている Layer 2 over Layer3 のカプセル化方式である。layer 2 over Layer3 のカプセル化方には L2TPv3 や VXLAN、Stateless Transport Tunneling(STT) など複数の方式が提案されているが、Geneve は Layer2 フレームだけではなく、それに関するメタ情報を付与できるよう設計されている。図 5.4 は、Geneve のヘッダ構造を示している。geneve ヘッダには、OAM フレームを示す O フラグや、Critical フレームを示す C フラグ、仮想ネットワークの識別子を示す VNI フィールドなどを持つ。そして、Protocol Type フィールドで、カプセル化した元フレームがどのようなプロトコルであるかを示す。2016 年現在、ここには 0x6558 で表される Ethernet Bridgeing のみが定義されている。geneve ヘッダの後ろには、Type/Length/Value(TLV) 形式で表現される Option フィールドを付与することができる。図 5.5 に示すとおり、Option フィールドは可変長であり、カプセル化したフレームに関するメタ情報を自由に拡張可能である。asteroid では、無線フレームを示すタイプを Geneve の Protocol Type に格納し、ハードウェアエミュレータの実装依存となる情報を Meta Data として Option フィールドに格納した。

受信時には、最初にカプセル化の際に付与したオプションを調べる。そして、メタデータを読み取り、フレームタイプが TX であれば、TX_ACK の送信を行う。その後、オリ

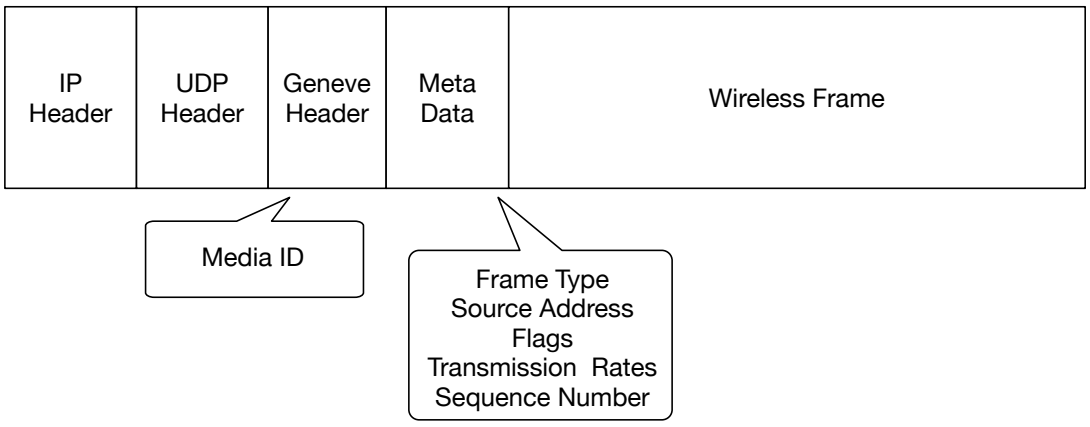



図 5.3: カプセル化のフォーマット

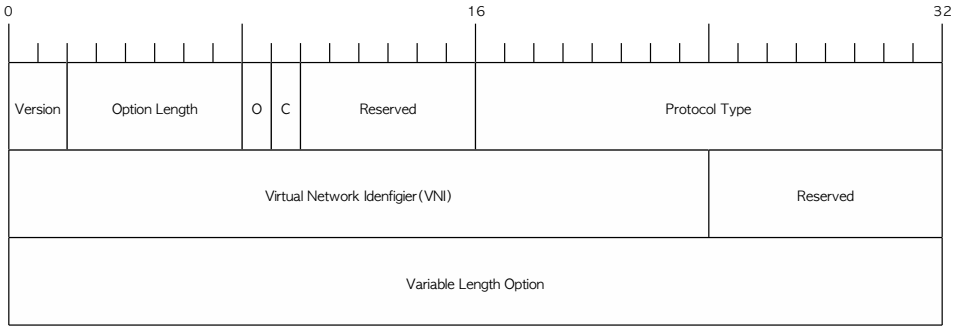



図 5.4: Geneve のフォーマット

ジナルの無線フレームを取得し、仮想無線インターフェースへと入力する。

### 5.5.3 無線フレームの送信時間の再現

mac80211_hwsim は、無線フレームの送信時に変調方式や送信データの長さを考慮した制御を行わないため、Mac80211_hwsim が送受信する無線フレームは、非常に短い時間で交換される。そのため、Asteroid を用いて有線ネットワーク上で無線フレームの交換を実現したとしても、無線ノードは無線インタフェースの設定に関わらず有線ネットワークと同じ回線速度となる。しかし、IEEE 802.11 には変調方式や転送レートなどの制御があり、アプリケーションソフトウェアはこの結果を受けて動作を決定する。

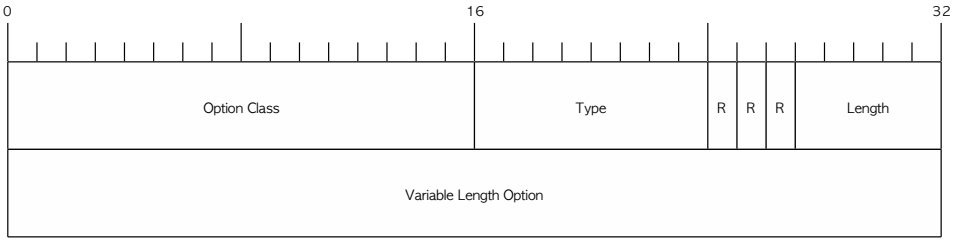



図 5.5: Geneve の Option フォーマット

図 5.6 は、フレーム送信時間の計算処理を示した図である。asteroid は、ハードウェアエミュレータ、Ethernet の双方向から受信したフレームに対して、送受信に必要な時間を計算する。無線フレームには、プリアンブルの長さやビーコンやプローブなどを行う管理フレーム、RTS/CTS、Acknowledgement などを示すフレームタイプ、データフレームのような種類がある。asteroid は、ハードウェアエミュレータから無線フレームを受信した際にフレームタイプの確認し、どのような処理を行うかを判断する。そして、無線フレームから得られた情報を元にフレーム送信時間の計算を行う。この時、Asteroid は送信処理中として lock フラグを立てる。lock フラグが立っている場合、Physical Interface からのフレームを受信したとしてもその場での処理は行わない。asteroid は、フレーム送信時間の時間待機した後、Ethernet へ送信処理を行い、lock フラグを倒す。これにより、無線インターフェースのような送受信を同時に行わない処理を実現する。

## 5.6 Asteroid の性能計測

本節では、Asteroid の性能について述べる。

性能評価を行うにあたり、StarBED の Group K と I を使用した。starBED が提供している Group I、K の性能は表 5.3 の通りである。また、OS は Ubuntu 16.04 を使用した。

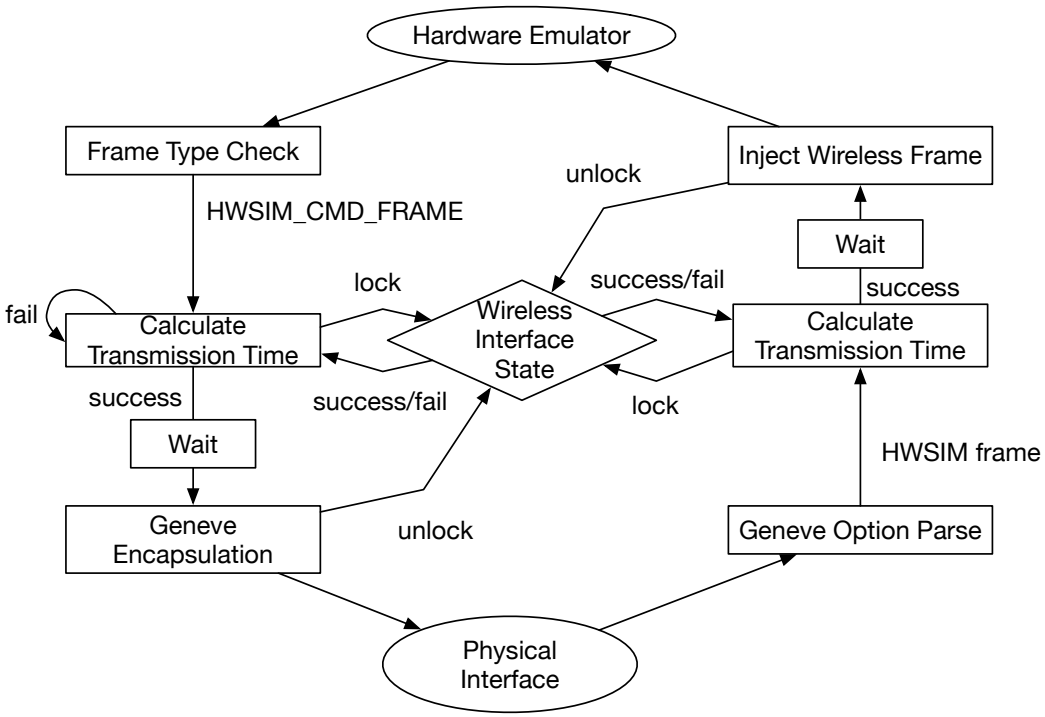



図 5.6: 無線フレームの送受信制御

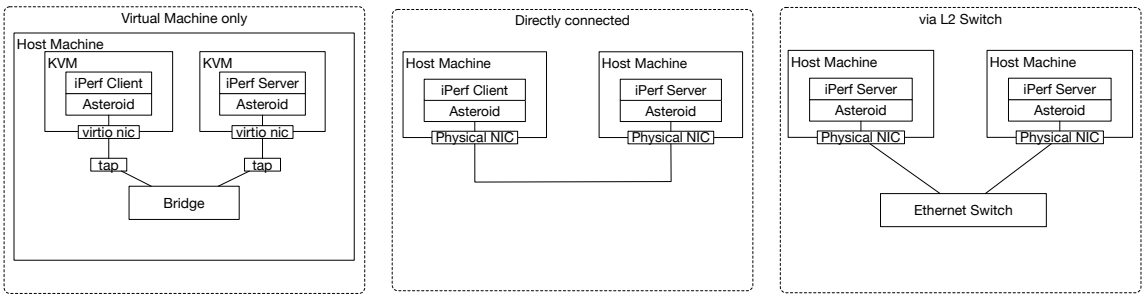
表 5.3: Server Specifications for Groups I and K

| | |
|---|---|
| Chassis | Cisco UCS C220 |
| CPU | Intel Xeon X5670@2.93GHz x4 |
| Memory | 48GB |
| HDD | 500GB x2 |
| NIC | Broadcom Gigabit Ethernet Quadport |
| OS | Ubuntu 16.04 Server AMD64 |

### 5.6.1 Asteroid の性能評価

#### スループット

asteroid の性能評価として DSSS と OFDM の 2 つの変調方式と対応する転送レート毎のスループットを計測した。計測環境は、図 5.7 の 3 通りである。




(a) Virtual Machine (b) Directly Connected (c) L2 Switch

図 5.7: スループット計測の環境

図 5.7a は、1 台のホスト OS 上で 2 台の仮想マシンを KVM を用いて動作させた。仮想マシン間は Linux bridge にて接続されており、計測環境の中では最も通信レイテンシが低くなる構成である。次に、図 5.7b は、2 台の物理マシン間を直接接続した構成である。最後の図 5.7c は、物理マシン間にイーサネットスイッチを設置した構成である。また、Asteroid は、無線ノード間の距離は再現しないため、この環境での無線ノード間の距離は 0 である。

最初に、インフラストラクチャモードでのスループットの計測を行った。図 5.8 は、DSSS と OFDM で提供される各リンク速度で計測を行った結果である。54Mbps でのリンク速度では、仮想マシン間での帯域が 20Mbps 以上となっており最も高速な結果となっている。次に、directly-connected で約 17Mbps となっており、L2 Switch を経由した結果が 14Mbps と最も低速な結果となっている。これは、計測環境の通信レイテンシが異なることから、Asteroid が 1 秒あたりに無線フレームを送信可能な回数に違いがでるためである。asteroid は、フレーム送信時間の計算を行い無線通信でのタイムスロットを模倣しているが、通信遅延を考慮して計算していないため通信遅延が大きくなるとフレームの送信タイミングを失うことになる。仮想マシン同士での通信では、ハイパーバイザ内の bridge interface のみ経由するため物理マシン間との通信と比較して遅延時間は小さくなるため最も高速に通信できている。しかしながら、リンク速度が 24Mbps 以下の場合、仮想マシン間の計測結果が大きく低下している。図 5.9 は、Asteroid 動作時の CPU 使用率である。この結果から、仮想マシンの CPU 使用率が通信レイテンシに影響

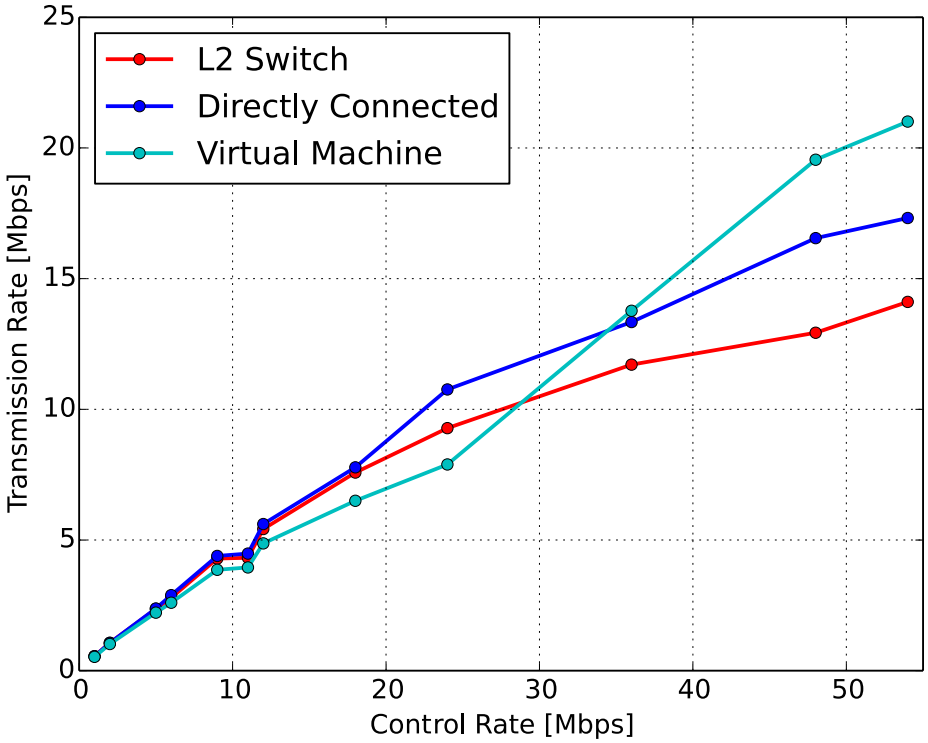



図 5.8: 転送レート毎の TCP スループット

していることがわかる。この通信レイテンシの影響は大きく CPU 使用率が数 % では、仮想マシン間の通信に 450$\mu$s のレイテンシが発生するが、CPU 使用率が 20 % を超えた場合、200$\mu$ を下回る結果となっている。

### 5.6.2 スケーラビリティ

次に、無線ネットワーク内のノード数を増加させた場合のスループットの計測を行った。計測は、図 5.10 で示すようにインフラストラクチャモードアドホックモードの 2 通りを構成し、複数の無線ノードが参加しているネットワーク上で 1 対 1 の通信 (2 Nodes) を行った場合、1 対他の通信 (All Nodes) を行った場合の計 4 通りの計測を行った。アドホックモードは、IEEE で 11Mbps のリンク速度と使用すると決められているため、本計測でも 11Mbps を使用している。インフラストラクチャモードは、IEEE 802.11g の最大リンク速度である 54Mbps を使用した。

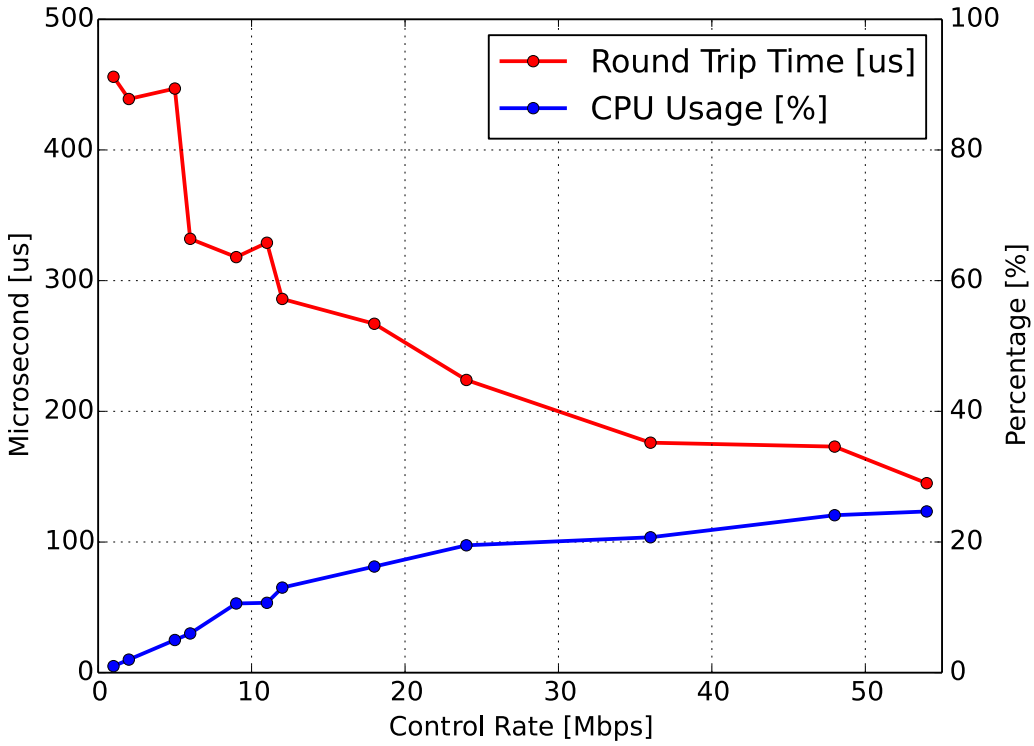



図 5.9: 制御帯域 v.s. CPU + Latency

図 5.11 は、無線ネットワーク内に複数の無線ノードを参加させた場合のスループットを示している。ノード数が 2 の場合、2 Nodes、All Nodes ともに同じ計測環境となるため、結果も同じとなっている。しかし、ノード数の増加とともに全ての計測で Transmission Rate が低下していることがわかる。アドホックモードの 2Nodes の結果では、4 ノードまでは帯域幅の低下が見えづらいが、5 ノード以降徐々に Transmission Rate が低下している。無線ネットワークでは、アプリケーションが行う通信以外にも Probe Request や Rekeying のような管理フレームの通信が行われている。asteroid では、これらの管理フレームも扱うためノード数が増加すると、フレームの衝突が発生する確率が高くなる。その結果、無線ネットワークに参加するノードが増加すると、たとえ 1 対 1 の通信のみが行われている状況でも利用可能な帯域幅は減少する。all Nodes では、これに加えて全てのノードが通信を行うため帯域の競合が発生し、Transmission Rate がより大きく低下している。

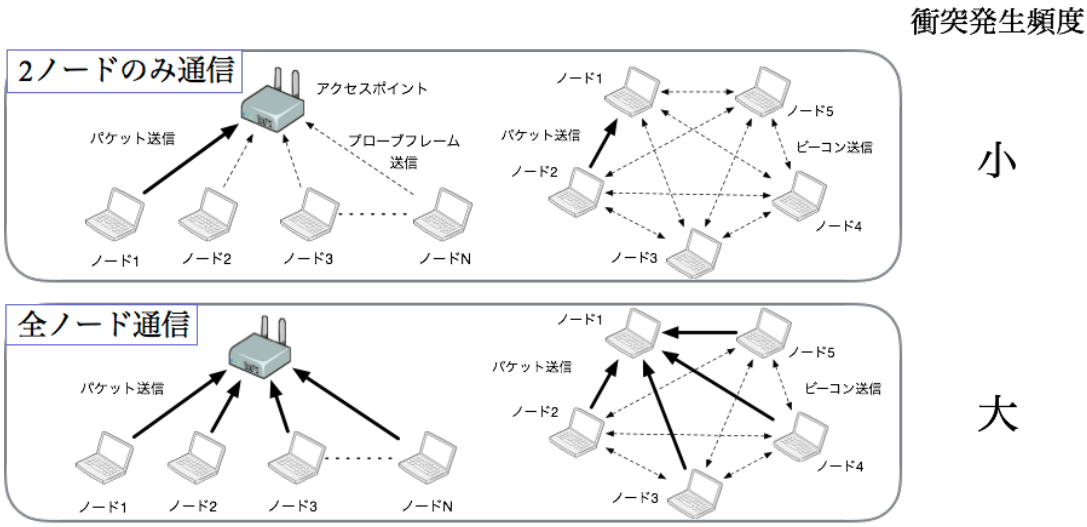



図 5.10: スループット計測環境 (衝突あり)

## 5.7　議論

### 5.7.1　遅延補正

asteroid は、無線フレームの送信に必要な時間を計算するが、有線ネットワーク上の遅延時間は考慮していない。そのため、Asteroid を動作させた環境によってスループットが変わってしまっている。本研究で評価に用いた StarBED の環境でのサーバ間の遅延時間は、約 100 $\mu$s であった。対して、1500Byte の無線フレームを送信するために必要な時間は 252 $\mu$s であるため、この遅延時間は、無線フレームの送信時間を計算する上で非常に大きな値となる。

この差分を補正するためには、Asteroid の処理時間、計算機間での通信遅延をフレーム送信時間へ含める必要がある。本研究での評価では、802.11g 54Mbps での通信時、約 14Mbps の帯域を利用可能である結果が出ているが、フレーム送信時間の計算結果から通信遅延である 100$\mu$s を引いた場合、約 20Mbps まで帯域幅が上昇した。しかしながら、この補正手法は非常に簡単なものであり、実際の通信では計算機の負荷により遅延時間は揺らぐため、状況に応じて補正し続ける仕組みが必要である。

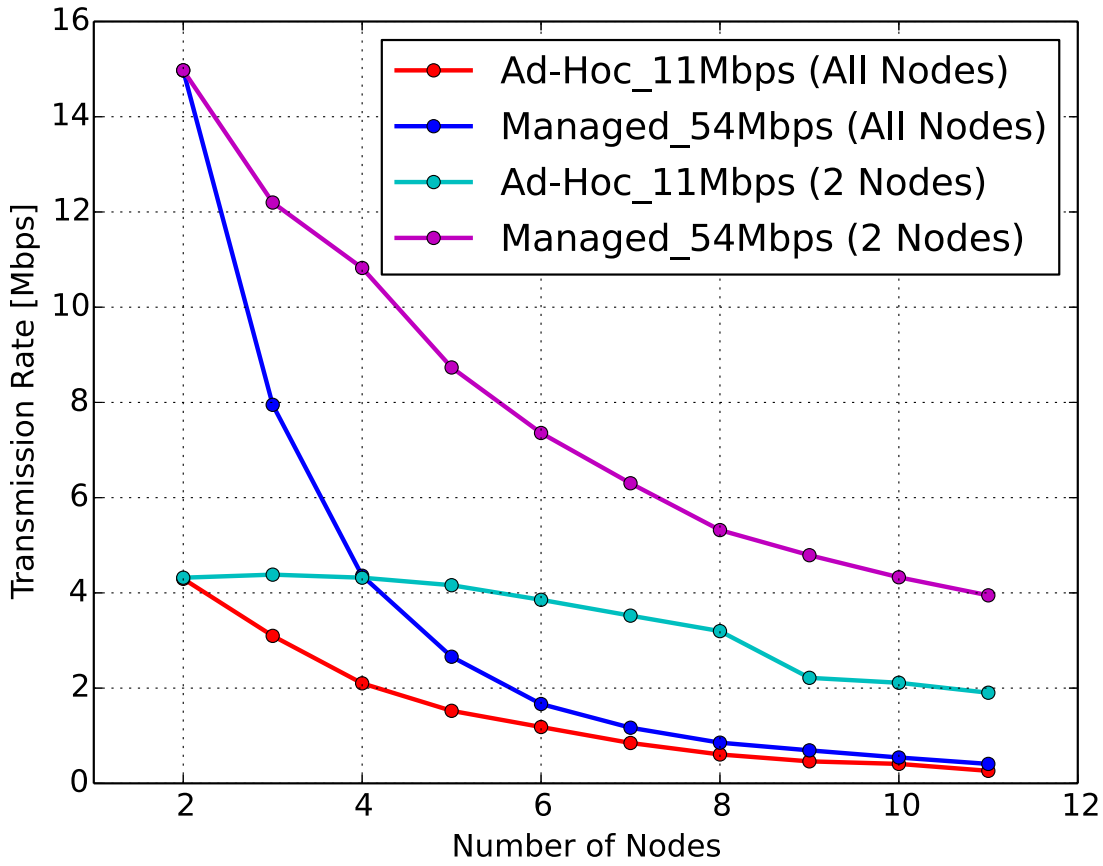



図 5.11: ノード数によるスループットの変化

### 5.7.2 チャネル干渉の再現

WLAN の 2.4GHz 帯で 1 から 13 channel まで扱える。そして、5 channnel 離れていればお互いの電波は干渉しないと言われている。これまでの伝搬エミュレータでは、シミュレーションで電波干渉を計算していた。Asteroid では、WLAN interface の channel 情報は取得しているが、channel の干渉までは制御していない。Asteroid は、WLAN フレームを伝搬させるソフトウェアであるため、channel 干渉も再現する機能が必要であると考えられる。

Asteroid でチャネル干渉を再現するためには、各フレームが使用している周波数を読み取り、同じ時間に到達した電波との影響を計算する必要がある。これに加えて、マルチパスやフェージングなどを考慮すると電波の反射や経路長を計算する必要がある。しか

しながら、無線フレーム、インタフェースから受信点までの反射や電波の経路長はわからないため、シミュレーションと組み合わせて動作する必要がある。これら全てに対応すると計算量が膨大となりリアルタイムでの処理が現実的ではなくなってしまう。そのため、Asteroid がリアルタイムでの処理を行い、かつチャネル干渉を再現するためには、Asteroid がチャネル干渉の計算を行うのではなくレイトレーシングのような伝搬シミュレーション結果からチャネル干渉による減衰、フレームロスの値のみを取得し、その結果をハードウェアエミュレータへ伝搬する方法が考えられる。

### 5.7.3 IEEE 802.11n/ac のサポート

本研究では、IEEE 802.11a/g による無線通信を Asteroid で実現したが、IEEE 802.11 には 11n や 11ac のようなより高速な無線ネットワークが広く使われている。asteroid が 802.11n や 802.11ac をサポートするには性能面で問題がある。asteroid は送信時間の計算を行わない場合に最も高い性能を出すことが可能となるが、この場合でも約 140Mbps の性能となる。asteroid が 802.11n や 802.11ac をサポートするためには、802.11n の理論値である 300Mbps 以上の性能を出せなければならない。現状の Asteroid のボトルネックは、mac80211_hwsim カーネルモジュールと Asteroid 間の通信部分である。そのため、より高速な Netlink Mmap を用いることで 802.11n や 802.11ac への対応は可能であると考えられる。

### 5.7.4 他のメディアのサポート

本研究では IEEE 802.11 による無線ネットワークを有線ネットワーク上に構築するため Asteroid を設計・実装した。しかし、無線ネットワークには IEEE 802.11 だけではなく Bluetooth や Zigbee なども使用される。IoT デバイスの検証を考えた場合、Asteroid はこれらの通信メディアも扱える必要がある。

Bluetooth は、ハードウェアエミュレータとして btvirt が BlueZ に含まれている。btvirt は phy のエミュレーションをユーザランドプロセスで行っており、プログラム内

に Bluetooth フレームである HCI コマンド、HCI イベント、ACL データ、SCO データを hook するための仕組みが提供されている。そのため、この hook 関数から Asteroid へフレームを転送することで、有線ネットワーク上で Bluetooth ネットワークを構築することが可能である。

IEEE802.15.4 は、ハードウェアエミュレータとして fakelb がカーネルモジュールとして提供されている。fakelb は、mac80211_hwsim と同様にカーネルモジュールで phy のエミュレーションを行っているため、mac80211_hwsim と同様に phy 間で送受信しているデータを Asteroid へ転送可能とすることで、有線ネットワーク上で IEEE 802.15.4 のネットワークを構築可能であると考えられる。

Asteroid は様々な通信メディアを扱えるようメタコンテナにメディア識別子を持つ。このメディア識別子を扱うことで他の通信メディアのサポートを容易に行うことが可能となる。

## 5.8　おわりに

本節では、有線ネットワーク上に仮想的な無線ネットワークを再現する Asteroid の設計と実装について述べた。ハードウェアエミュレータは、汎用 OS 上で無線インタフェースや小型デバイスをエミュレーションすることで、無線空間を想定した疑似環境内での検証が可能である一方、規模追従性に問題があった。擬似的な無線空間を再現する中で、実際の無線デバイス、インタフェースが持つ様々な機能をアプリケーションソフトウェアに提供することは、忠実性の観点から非常に重要である。そこで本研究で提案する Asteroid は、ハードウェアエミュレータから送信された無線フレームを有線ネットワーク上で通信可能とすることで、これまでに実現不可能であった規模の無線ネットワークを構築可能とした。

# 第 6 章

# NETorium

## 6.1 はじめに

これまでの無線ネットワークエミュレーション手法は、無線伝搬エミュレーションとハードウェアのエミュレーションのどちらかを選択していた。QOMET や Mobinet のような無線伝搬エミュレーションは、高いスケーラビリティと遅延、帯域幅、パケットロスの模倣が実現できる。一方で、ハードウェアエミュレーションは、OS やアプリケーションソフトウェアから RSSI や SSID のような無線に関する情報の取得、利用ができる特徴がある。しかし、我々の生活に使用しているネットワークや、公衆無線ネットワークサービスでは、ユーザ認証が行われ、数百といったユーザが存在する。また、震災を想定した耐災害 ICT ネットワークでは、より多くのユーザを支えるネットワークを構築する必要がある。

そこで本研究では、遅延、帯域幅、パケットロスのエミュレーション及び、ハードウェアエミュレーションを NETorium で実現した。個別ではできていた伝搬エミュレーションとハードウェアエミュレーションのメリットを両立させることで、より忠実度の高いネットワークエミュレーションが可能となる。また、NETorium は汎用 OS 上での動作が可能であるため、実際のプログラムコードを用いたテストが可能である。

NETorium は Meteor と Asteroid から構成される高忠実な大規模無線ネットワークエミュレータである。Meteor はスケーラブルかつ低レイヤで動作する伝搬エミュレータと

して設計・実装を行った。これまでの伝搬エミュレータでは不可能であった IPv6 やデータリンク層のプロトコルに対応し、より大規模な無線環境の再現を可能である。Asteroid はハードウェアエミュレータを他のノード間で通信可能とする仮想無線ネットワークエミュレータである。ハードウェアエミュレータから送信される無線フレームを有線ネットワーク上で通信可能とすることで、無線に関する情報を取得しつつ、複数サーバでの無線ネットワークエミュレーションを可能とした。

## 6.2 高忠実大規模無線ネットワークエミュレータ

### 6.2.1 伝搬エミュレータとハードウェアエミュレータの結合

既存研究で提案されている伝搬エミュレータは、スケーラビリティをもちつつ無線空間の伝搬特性の再現が可能であった。一方で、ハードウェアエミュレータでは、高い忠実度が実現できる。これら 2 つの手法は、機能面においてトレードオフの関係がある。しかし、実際の無線ネットワークでは、認証を用いたり、無線ネットワークの状態を考慮してトポロジを構築するものがある。そして、ノードの数も多数存在し、複雑なネットワーク構成となる。より忠実かつ、大規模な無線ネットワーク環境を再現するためには、 2 つのアプローチのメリットを両立させる必要がある。

本研究では、伝搬エミュレータである Meteor [39, 43] とハードウェアエミュレータを結合する Asteroid から構成される無線ネットワークエミュレーション環境 NETrium を提案する。仮想無線ネットワークは、有線ネットワーク上に構築するハードウェアエミュレータ間での通信を可能とする擬似的な無線空間である。

Meteor は、ネットワーク層のプロトコルを意識せず、簡素にマルチホップ環境でも無線空間を再現できる伝搬エミュレータである。Asteroid は、ハードウェアエミュレータから送信される無線フレームを有線ネットワーク上で通信可能とするソフトウェアである。図 6.1 は、Meteor と Asteroid から構成される Netorim 環境を示した図である。Ethernet で接続されたサーバ上で動作するハードウェアエミュレータは、Asteroid による仮想無線ネットワークにて channel 毎に分類される。Asteroid によって構成される

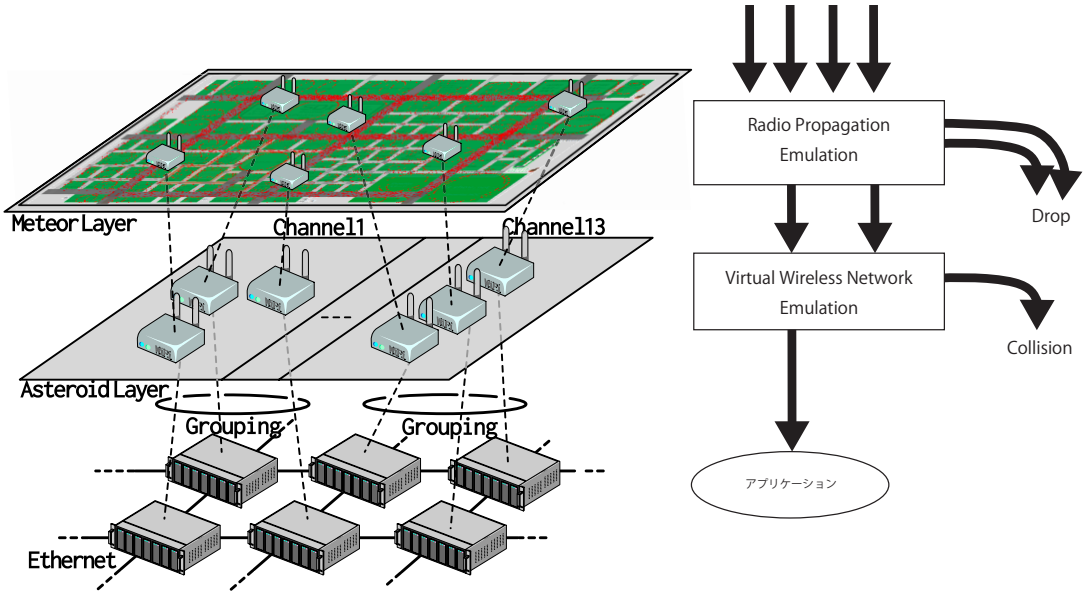



図 6.1: NETorium のデザイン

Asteroid Layer では、無線フレームによる通信を Ethernet 上で実現する。そして、各ノード間の距離や移動、そして到達性を Meteor Layer で再現する。有線ネットワークはバス型トポロジで構築されているため、有線ネットワーク上には全てのノードのトラフィックが送信される。このトラフィックを Asteroid が処理してしまうと、無線フレームの処理を行う負荷が大きくなってしまう。

Netorium では、まず他のノードから受信した無線フレームは Meteor による伝搬エミュレーションで処理される。ここで、到達不可能な位置から送信された無線フレームは破棄される。これにより、Asteroid は到達性のあるノードからの無線フレームのみを処理することが可能となる。そして、Meteor を通過した無線フレームは、次に Asteroid で衝突判定が行われ、衝突と判定された無線フレームは破棄、通過した無線フレームはハードウェアエミュレータを介してアプリケーションへ渡される。

Meteor と Asteroid を組み合わせることで、有線ネットワーク上に伝搬特性を再現した上での無線ネットワークを構築することができる。これにより、これまでできなかった WPA2 や hotspot のような認証や 802.11s [44] のようなメッシュネットワークの構築、その上でのアプリケーションテストが可能となる。

## 6.3 評価

### 6.3.1 ノード間の距離毎のスループット

本節では、NETorium を用いて 2 台の無線ノードの位置によって変化する帯域幅を計測した。トラフィックの生成には iperf を使用し、無線ノードは IEEE 802.11g のアドホックモードで接続した。無線ノードは、距離が徐々に開いていくシナリオとした。図 6.2 は、Meteor と Asteroid を動作させた上で通信帯域を計測した結果である。2 ノード間の距離は 10m 単位で計測し、各距離におけるシナリオの帯域幅、計測結果の帯域幅、シナリオのパケットロスレートをプロットしている。

DeltaQ のシナリオでは、120m から徐々に帯域幅の低下が見られているが、計測結果では変化が見られない。これは、deltaQ による帯域計算が Asteroid での通信帯域よりも大きいためである。2 ノード間の距離が 150m になると、deltaQ の計算帯域が低下するため、Actual rate の結果も低下している。160m 以降では、パケットロスレートが 100% になっているため、Actual rate では、通信不可能な状態になっている。deltaQ の帯域幅が 1Mbps 程度となっているのは、deltaQ が扱うパケットロスレートが 100% となっているため、問題ないと考えられる。

### 6.3.2 規模追従性

スケーラビリティの評価として 1000 台規模の無線ネットワークを構築し、通信トラフィックの計測を行った。構築した無線ネットワークは、160 x 100 のフィールド内にアクセスポイントを 32 台、ステーションを 1000 台、そして HTTP サーバを 1 台用意した。各アクセスポイントとステーションの配置を図 6.3 に示す。各アクセスポイントに接続したステーションは、8 から 54 である。ステーションは、フィールド内を移動するユーザのシミュレーション行い、その出力結果を用いた。図 6.4 は、各ステーションがアクセスポイントを経由してサーバとの通信を行った際のスループットを示している。ステーション数が 8 の場合、平均 70Kbps のスループットが出ており、40 台以上になると

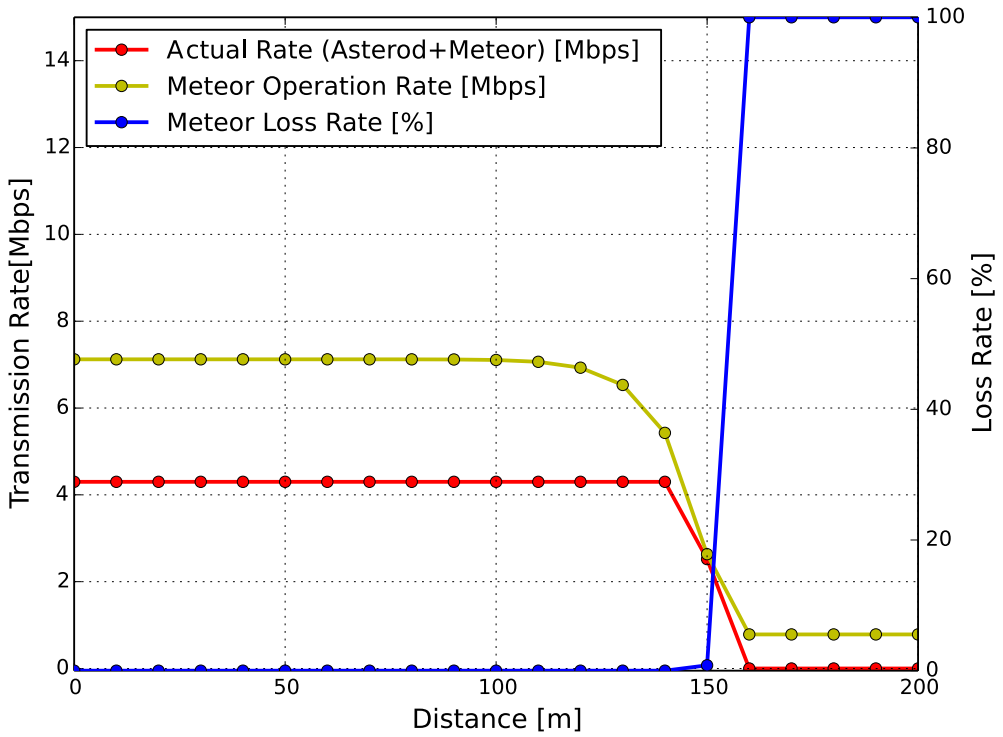



図 6.2: 2 ノード間の距離によるスループットの変化

平均 2Kbps 程度のスループットとなっている。20 から 30 台のステーション数の結果を見ても、アクセスポイントに接続しているステーションの数によって、スループットが変化していることがわかる。ステーション数が 8 の時、23Kbps の結果も出ているが、これはアクセスポイントとステーション間の距離が長いため、平均スループットが低下しているものである。

## 6.4 議論

### 6.4.1 伝搬エミュレータの動作

これまでの伝搬エミュレータは、ノード間での遅延時間、利用可能な帯域幅、パケットロス率を再現することで、擬似的な無線空間を再現していた。しかし、本研究で提案した NETorium は、Asteroid による仮想無線ネットワークを提供する。そのため、Asteroid

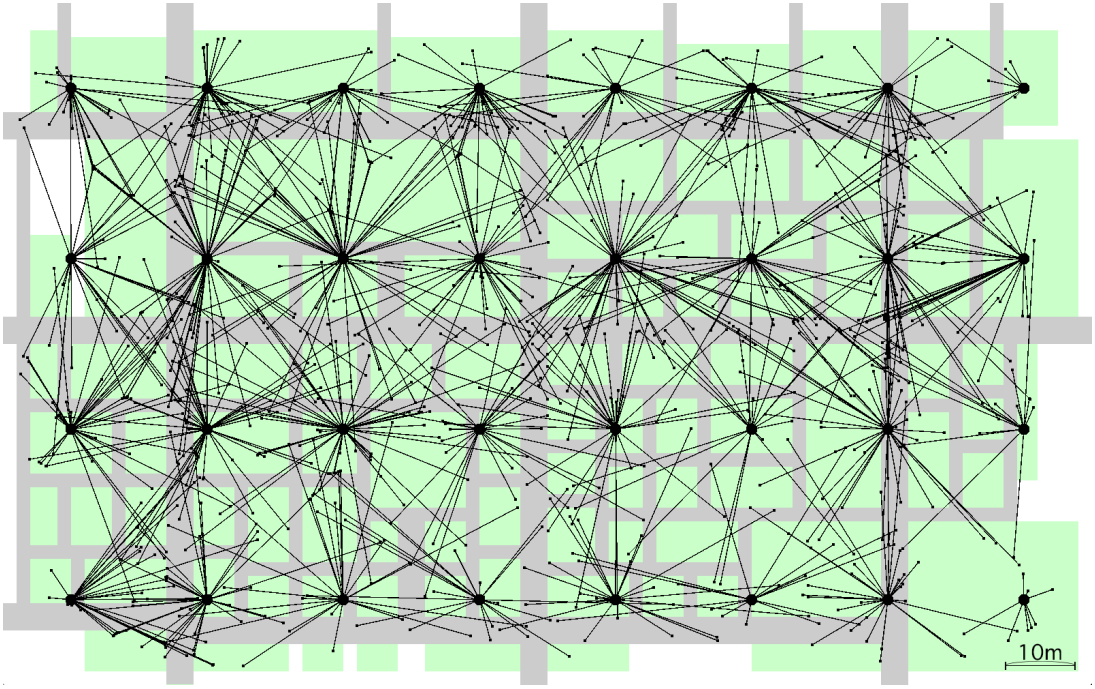



図 6.3: シナリオレイアウト: 32 アクセスポイント、1000 ステーション

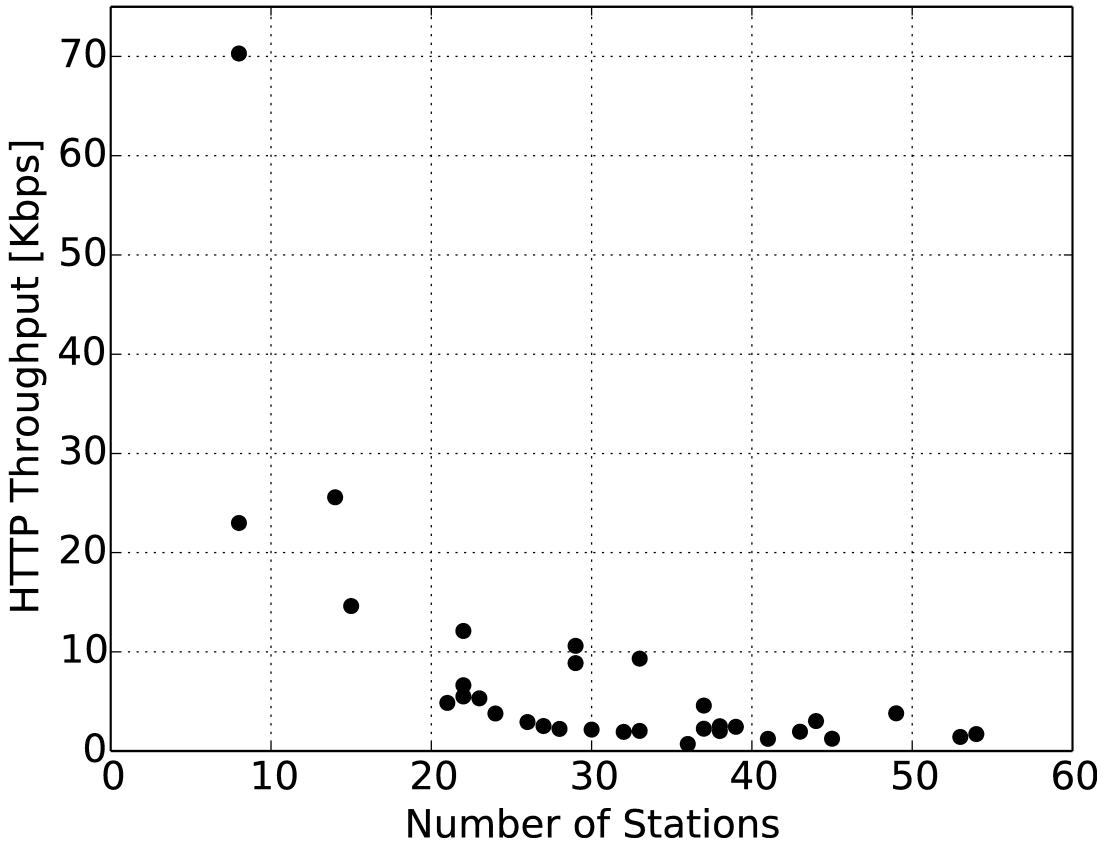



図 6.4: アクセスポイント毎の平均スループット

は、管理フレームや無線フレームの再送制御を行うため、伝搬エミュレータはフレームエラーを再現するだけで無線ネットワークの再現が可能であると考えられる。

伝搬エミュレータのスケーラビリティは、遅延時間、利用可能な帯域幅、パケットロス率を再現する処理時間に大きく依存する。その中でも、遅延時間と帯域幅の再現には、パケットの保持と送信時間の制御が必要となるため、CPU とメモリを消費する。これ対して、パケットロス率は受信時に確率に従いパケットを破棄するのみでよいため処理が簡素である。もし、無線ネットワークの再現を行うための処理がパケット破棄のみで十分となれば、より大規模な無線ネットワークの再現が可能となると考えられる。

### 6.4.2 RSSI の制御

実際の無線ネットワークでは、周辺環境によって RSSI の値は動的に変化するが、現状の Asteroid は、RSSI の値を固定値で扱っている。NETorium では、無線フレームの伝搬は Asteroid が行うが、空間伝搬に関する情報を扱うのは Meteor であるため、Asteroid は、空間伝搬パラメータである RSSI の値を持たない。

### 6.4.3 ブロードキャストトラフィックの制御

無線ネットワークではブロードキャストが多く使用されるが、有線ネットワークでは同じレイヤ 2 ネットワークに所属するため全ノードへの到達性を持ってしまう。Meteor や QOMET は、これに対して受信ノードでフレームの破棄を行うことで到達性の制御を行っている。しかし、この手法では全てのノードが送信したパケットを処理するため受信パケットを処理する負荷が大きいという問題がある。

展開する無線ノード数が 1000 台程度の規模の場合、受信時での処理でも実現可能である。しかし、より多数のノード、数千や数万といった大規模なノード数を展開した場合、フレームの受信および破棄の処理が他のアプリケーションソフトウェアへ影響が出てしまうほどの大きな負荷となってしまう。数万ノードを展開する大規模ネットワークをエミュレーションするためには、送信されるブロードキャストフレームに対して到達可能なノー

ドのみへフレームを配送する仕組みが必要である。本研究では、この問題について言及をしていないが、ブロードキャストフレームをマルチキャスト、またはユニキャストへ変換し、到達性のあるノードのみへフレームを配送することでネットワーク全体の負荷を削減できると考えている。

## 6.5 本研究成果の適用例

### 6.5.1 CGN のモバイルネットワークへの適用性検証

2011 年 4 月、APNIC が管理する IPv4 アドレスが枯渇し、IPv4 の後継規格である IPv6 への移行が望まれている。しかし、IPv6 に未対応なアプリケーションが数多く存在することや、セキュリティ上の観点から IPv4 の延命措置として ISP が NAT を行う Carrier Grade NAT(CGN) [45] の導入も進められていた。CGN は、ISP Shared Address の割当を受けたユーザ毎の公平性を保つことや悪意あるユーザの追跡が可能でなければならないなど、従来の NAT とは異なる考慮が必要である。そこで、CGN 環境下でのユーザの挙動を再現し、CGN の最適配置についての調査を行った。

CGN が導入されるネットワークは ISP だけではなく、モバイル通信事業者が展開するモバイルネットワークにも必要である。しかし、無線ネットワークは、有線ネットワークとは異なり安定性や信頼性が低い。特に、3G や LTE のようなモバイルネットワークでは、ハンドオフ時のネットワーク遅延、パケットロスが CGN の性能に影響を与える可能性がある。そこで、多数のユーザが存在するモバイルネットワークを StarBED 上で再現し、CGN の性能評価を行った。図 6.5 は、構築した実験環境である。

HTTP Server は、本検証用に実装した秒間コネクション確立数を計測するためのアプリケーションサーバである。クライアントには、本検証用に実装したプロセスベースの PC クライアントと、仮想化技術を用いて実行している Android を最大で 30 万ノード設置した。この HTTP Server とクライアント群の間に CGN を設置し、クライアント群と CGN の間にモバイルネットワークのエミュレーションを行うため Meteor を設置した。Meteor はクライアント群と CGN の間に透過的に設置しており、クライアント群か

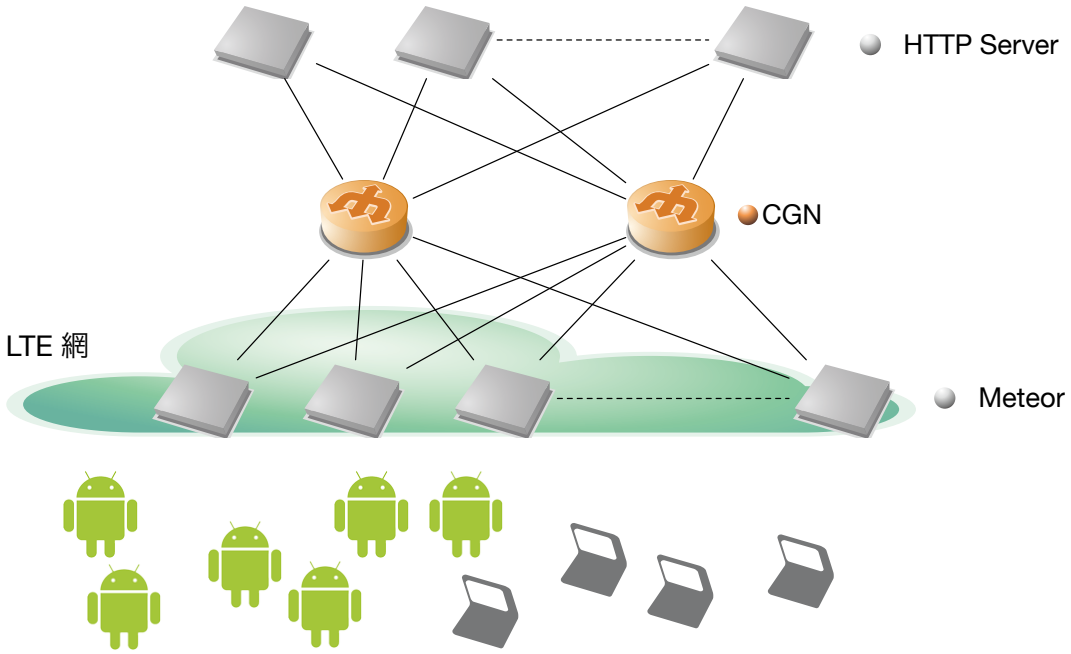



図 6.5: モバイル通信端末を収容する CGN 環境の再現

らは Meteor の存在は確認できない構成となっている。この環境で、クライアント群から HTTP Server へのアクセスを行うことで、CGN の性能限界を計測した。

### 6.5.2　仮想 Interop 無線ネットワークエミュレーション

Interop Tokyo 2016 のアカデミックパビリオン、北陸先端科学技術大学院大学ブースにて、1000 ノードを用いた IEEE 802.11s メッシュネットワークのデモを行った。本実験では、Interop 2015 の会場内を 1000 人の来場者が歩き回る状況を再現し、各来場者間で IEEE 802.11s を用いたメッシュネットワークを構築する。図 6.6 は、Interop Tokyo 2015 の会場マップ上で構築した無線メッシュネットワークの様子である。

## 6.6　おわりに

本節では、高忠実な無線ネットワークエミュレーションとして NETorium のデザインについて述べた。これまでの無線ネットワークエミュレータは、無線空間における伝搬特性を再現することで有線ネットワーク上で無線ネットワークの振る舞いを再現する伝搬エ

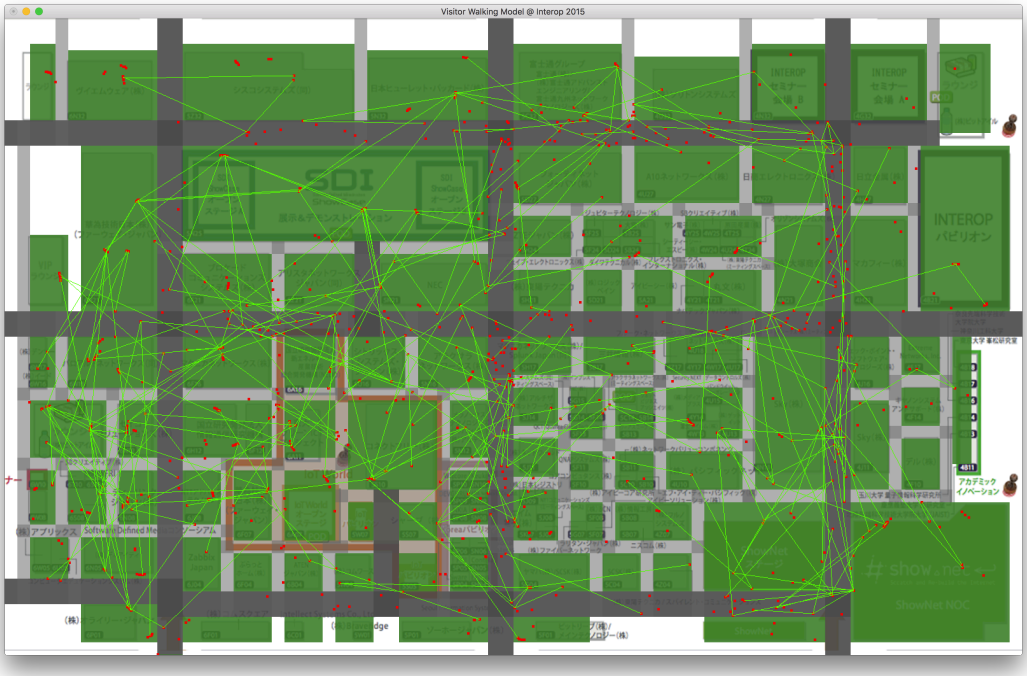



図 6.6: 仮想 Interop 無線ネットワークエミュレーション

ミュレータとサーバ上で無線フレームの通信を行うことで高忠実な無線ネットワークを構築するハードウェアエミュレータのどちらかであった。どちらも無線ネットワークエミュレータも無線ネットワークでの振る舞いを再現可能であるが、伝搬エミュレータは忠実さに、ハードウェアエミュレータは規模追従性に問題があった。そこで本研究では、高い規模追従性を持った伝搬エミュレータである Meteor と有線ネットワーク上で無線フレームによる通信を可能とする仮想無線ネットワークエミュレータである Asteroid を組み合わせることで両エミュレータの利点を持つ Netorium を提案した。Netorium を用いることで、これまで不可能であった無線プロトコル、メッシュ技術を用いた無線ネットワークを有線ネットワーク上に大規模に展開し、その上で様々なアプリケーションソフトウェアの検証を可能とした。

# 第 7 章

# より高度な無線ネットワークエミュレーションにむけて

我々の生活に強く密着した情報サービスは、サービスの品質次第で金銭的損害だけではなく身体への影響を及ぼす可能性がある。このような情報サービスは、ユーザへ展開する前に十分な検証を行い、想定外の動作を極力排除することが重要である。しかしながら、情報サービスの多くは、利便性の高い無線ネットワーク技術の上での利用が前提となっており検証が困難になってきている。一つの通信技術のみを用いている環境では、問題が発生しなくとも、 多種多様な技術が混在するネットワーク環境では、想定外の動作が起こる可能性もある。本論文では、このような複雑化する情報サービスと多様化するネットワーク環境について整理し、情報サービスの検証における課題について述べた。

## 7.1 より大規模な無線ネットワークエミュレーション

本論文で NETorium は、フルメッシュ型の無線ネットワークで 1,000 ノードの規模まで再現が可能であることを示した。しかしながら、10,000 台規模のノード数を展開した場合、OS のネットワークスタックがボトルネックとなり通信が不安定になる現象が発生する。無線ネットワークでは、有線ネットワークと比較して、管理フレームの通信が多く発生するため 1 秒あたりのフレーム数は多くなる。また、規格によって定められた電波帯

を共用するため、通信はブロードキャストで行われる。このような通信方式を有線ネットワーク上でエミュレーションするため、伝搬エミュレータでは受信時にフィルタリングを行い、ノード間の到達性を制御する。つまり、無線ネットワークエミュレーションを行う際には一度パケットを受信し、その後アプリケーションへの配送もしくは破棄を行うことで無線空間の振る舞いを再現している。そのため、無線空間上で通信可能なノード数に関わらず、無線ノードの数が増加すると 1 台のノードが受信するトラフィックも増加する。

より大規模な無線ネットワークを構築するためには、受信するトラフィックの低減手法が必要である。仮に、1 秒間に 50 パケット送信するノードを 10 万ノード展開した場合、無線ノードが 1 秒間に受信するパケット数は 500 万パケットである。これは、現在の Linux で処理できるパケット数の限界に近く、伝搬エミュレーションを行うには厳しい数値である。これを解決するためには、無線ノードが送信するブロードキャストを、シミュレーションで計算された到達可能な無線ノードにのみ配送するような仕組みが必要であると考えられる。

## 7.2　新しい通信メディアへの対応

NETorium は、本論文中で Bluetooth や Zigbee などへの対応が可能であると述べた。複数の通信メディアを有線ネットワーク上で通信可能とする Asteroid は、カプセル化を行う際にメディアの識別子を埋め込むことで、カプセル化された無線フレームをどのハードウェアエミュレータに渡すべきかを判断することができる。本論文では、MAC80211_HWSIM を用いた IEEE 802.11 のみの評価となっているが、Linux BlueZ に含まれている Bluetooth Emulator である btvirt を用いた Bluetooth ネットワークへの対応も可能であることがわかっている。また、Zigbee に関してもハードウェアエミュレータと無線フレームをフックする手段さえ確立すれば、Asteroid によるカプセル化を適用することで有線ネットワーク上での通信が可能である。

## 7.3 小型端末のエミュレーション

IoT が実現する世界は、様々な小型端末が情報の発信、交換を行いサービスを提供する。本論文では、インタフェースレベルでのエミュレータを用いて構築する仮想無線ネットワークを構築したが、IoT 向け小型端末はより特殊な用途で使われることが予想される。センサーデバイスやセンサーデバイスから情報を収集するような小型端末は、IoT 向け OS である Cooja と Asteroid が連携可能となることでネットワークテストベッド上に展開可能である。しかし、UAV や自律制御ロボットのような端末は、位置情報や気温、風速のような環境情報の影響を受けて動作する。NERVF では、UAV を利用した無線メッシュネットワークの構築を行っているが、UAV が指定された位置の周辺を飛行するシナリオとなっており気温や風速などの情報から動作を決定する機構は導入されていない。このような環境情報を考慮することで、より現実に近い動作がシミュレーション可能となるだろう。そして、シミュレーション結果を NETorium に反映させることで、より微細な変化も反映した無線メッシュネットワークの構築が可能となる。

## 7.4 街や人を含めた環境エミュレーション

複雑化する情報サービスは、街を構成する建造物、人の動きや密度によってその動作を変える。特に IoT のような様々なモノがネットワークに繋がるような環境が実現すれば、人の流れや密度によってサービス内容を変えるものもあるだろう。現在でも、Google Map のような経路案内サービスは、交通状況などを考慮し、混雑している経路を避けるようユーザに案内する。当然ながら、人が身につけるデバイスがネットワークに繋がり様々な情報を提供する環境が実現した場合、このような状況によって変化するサービスはより増加すると考えられる。情報サービスは、社会や我々の生活に密着するほど、周辺環境や動作に対して敏感になると言えるだろう。このような情報サービスは、サーバとネットワークだけでは不十分な検証となるため、より高忠実なエミュレーションが必要となる。本論文では、高忠実かつ大規模な無線ネットワークエミュレーションについて取り組

んだが、これだけで すべての環境を再現することは難しいだろう。実世界では、熱や風のような自然現象、電力や水のような生活を支える基盤など様々な要素が存在する。そして、我々はこの要素に少なくとも影響を受け、行動している。

本論文では、無線ネットワークを再現する手法を主眼においているが、アプリケーションソフトウェアの検証基盤として考えた場合、人が身につけるデバイスや各所に配置するような小型デバイスのエミュレーションが必要であると考えられる。そして、それらのデバイスは人の動作によって動作が変化するため、シミュレーションとエミュレーション、この 2 つの要素が相互に影響し合うようなフレームワークが必要となるであろう。

# 第 8 章

# おわりに

本論文では、様々なモノが繋がり社会を支える情報サービス、そして情報サービスを支えるネットワーク技術の検証を可能とする検証基盤を実現するべく、高忠実かつ大規模な無線ネットワークエミュレータの設計を実装を行った。情報サービスを支えるネットワークは、利便性や信頼性の向上を目的として様々な通信技術が提案されている中、ネットワークテストベッド上でこれらの技術を用いて検証を行うための手法、必要となる要素について議論を行い、その概念を実証した無線ネットワークエミュレーション基盤を提案した。

人間社会の基盤として利用されている情報サービスは、技術の発展とともに利用場面は多岐にわたる。そして、情報サービスを支えるネットワークも、いつでも、どこでも使えるよう無線ネットワーク技術が導入されている。利便性を追従した情報サービス、ネットワークは、様々な技術が導入され複雑な構造となっている。このような環境で提供される情報サービスの検証は難しく、想定外の事象が発生することも多い。ネットワークテストベッドは、あらゆる事象を再現し、情報サービスやそれを駆動させるアプリケーションソフトウェアの検証の場として使われているが、複雑化するネットワーク環境への追従が難しくなってきている。本論文では、この複雑化するネットワーク環境の中でも、無線ネットワーク技術に焦点を当て、より高忠実な無線ネットワークエミュレーションに必要となる要素について議論した。そして、ネットワークテストベッド上に仮想的な無線ネット

ワークの構築、大規模かつネットワークプロトコル非依存な伝搬特性の再現を実現する Meteor と Asteroid の実装を行い、多種多様な無線ネットワーク技術を高忠実に再現した上でのアプリケーションソフトウェア検証が可能であることを示した。

# 謝辞

本研究を進め、論文を執筆するにあたり、終始御指導賜りました篠田 陽一教授に深く感謝致します。また、本学 丹 康雄教授、知念 賢一 准教授、BEURAN, Razvan Florin 准教授ならびに名古屋大学 河口 信夫教授には、審査委員をお引き受けいただき、多くの助言を頂けたことを深く感謝致します。

次に、本学 篠田研究室のスタッフに感謝したい。本研究室の 宇多 仁 助教 には、研究に関して多大なご指導を賜りました。また、研究環境構築研究、実験等様々な方面に関して多大なご指導を賜りました。ここに深く感謝いたします。

また、研究室の修了生、現メンバーにも感謝します。特に、井上 朋哉博士、安田 真悟博士、高野 祐輝博士には、研究と学生生活の両面で大きな支援をもらった事を感謝します。また、先輩、同期、後輩の LATT Khin Thida 博士、 Nguyen Lan Tien 博士、NGUYEN Nam Hoai 博士、Muhammad Imran Tariq 氏、松井 大輔氏、千装 俊幸氏、佐川 喜昭氏、栗原 良尚氏、川瀬 拓哉氏、立花 一樹氏、中村 祐輔氏、橋本 将彦氏、山田 悠介氏、吉岡 慎一郎氏、鍛冶 祐希氏、村上 正太郎氏、田部 英樹氏、向井 雄一郎氏、大野 夏希氏、向井 康貴氏、岩本 裕真氏、加藤 邦章氏、岩橋 紘司氏、成田 佳介氏、園田 真人氏、八木 辰弥氏、可児 友邦氏、廣瀬 真人氏、三木 晶司氏、押川 侑樹氏、橋本 光世氏、村上 正樹氏らと共に学生生活を歩めたことを感謝したい。

次に、本研究の無線ネットワークエミュレーション手法および実験を行う上で支援いただいた方々に感謝したい。情報通信研究機構北陸リサーチセンターの三輪 信介博士、宮地 利幸博士、太田 悟史氏、中井 浩氏には StarBED での実験において多大なご助力頂けましたこと心より感謝いたします。Panasonic 株式会社 村本 衛一氏、小西 一暢氏、石

井 秀教氏には、本研究を行う上で様々な助言を頂き非常に感謝しています。また、NTT コミュニケーションズ株式会社 小原 泰弘氏には、研究面だけではなく、生活面でも大変お世話になりました。株式会社 Clwit の井澤 志充博士、清原 智和氏に感謝したい。日々の仕事を御一緒させて頂けたことは大変刺激になりました。さらに、WIDE プロジェクトの方々に方々に感謝したい。同じネットワークの研究をする仲間として交流できたことは、たいへん大きな刺激となりました。以上、全ての方の温かいご支援がなければ本論文を完成させることは不可能でした。心から感謝いたします。

最後に、研究や生活を支えてくれた家族、父・志郎、母・光子、妹・麻里には本当に迷惑をかけました。心から感謝を表し謝辞と致します。

# 本研究に関する発表論文

論文誌 (査読有り)

- 明石邦夫, 井上朋哉, ラズバン ベウラン, 篠田陽一. Meteor: 大規模ネットワーク実験環境における無線ネットワークエミュレータの設計と実装. ネットワークソフトウェア技術とその応用論文特集. Vol.J98-B, no.4, pp. 357-372, Apr. 2015.
- Shingo Yasuda, Kunio Akashi, Masatoshi Enomoto, Shinsuke Miwa, Yoichi Shinoda, “Technical Requirements of Experiment Support Software for Huge-Scale Network Environment”, International Journal of Computers and Communications,id.123,2012.
- 村本 衛一, 香林 誠, 石井 秀教, 明石 邦夫, 知念 賢一, 篠田 陽一. 無線網に対応した実時間映像音声伝送制御の検証のための時間推移ネットワークエミュレーションとその用法. 理論・実践に立脚したインターネットアーキテクチャ論文特集. Vol.J98-B. no.10 pp. 1060-1069. Oct. 2015.

国際会議 (査読有り)

- Kunio AKASHI, Shingo YASUDA, Tomoya INOUE, Yuki TAKANO and Yoichi SHINODA, NETorium: High-Fidelity Scalable Wireless Network Emulator, AINTEC ’16, pp 25 - 32, November, 2016.
- Thilmee Baduge, Boon Ping Lim, Kunio Akashi, Jason Yew Beng, Ken-ichi Chinen, Ettikan Kandasamy Karuppiah, and Eiichi Muramoto. Functional

and performance verification of overlay multicast applications a product level approach. IEEE Consumer Communications and Networking Conference 2010, Jan 2010.

- Shingo Yasuda, Kunio Akashi, Tomoya Inoue, Toshiyuki Miyachi, Shinsuke Miwa, Ken-ichi Chinen, Yoichi Shinoda, "Requirements of Large Data Distribution Mechanism for Large-Scale Network Testbed", Proceedings of the 2nd European Conference of Computer Science (ECCS '11), ISBN:978-1-61804-056-5, pp.315-322, Tenerife, Spain, Dec, 2011.
- Toshiyuki Miyachi, Razvan Beuran, Yoshiki Makino, Kunio Akashi, Shingo Yasuda, Tomoya Inoue, Shinsuke Miwa, Satoshi Uda, Yasuo Tan, Yoichi Shinoda, "On network system evaluation under various failures", VALUETOOLS 2012: 226-227.
- Yuuki Takano and Ryosuke Miura and Shingo Yasuda and Kunio Akashi and Tomoya Inoue, "SF-TAP: Scalable and Flexible Traffic Analysis Platform Running on Commodity Hardware", 29th Large Installation System Administration Conference (LISA15), USENIX Association, pp.25-36, Nov 2015.
- Shingo Yasuda, Kunio Akashi, Toshiyuki Miyachi, Razvan Beuran, Yoshiki Makino, Tomoya Inoue, Shinsuke Miwa and Yoichi Shinoda, "Emulation-based ICT System Resiliency Verification for Disaster Situations", Workshop on Resilient Internet based Systems (REIS 2013), to be appeard,SITIS 2013, Kyoto, Japan, 2-5 December 2013.
- Yasuhiro Ohara, Kaname Nishizuka, Chinen Ken-ichi, Kunio Akashi, Makoto Kohrin, Eiichi Muramoto, Shin Miyakawa. On the Impact of Mobile Network Delays on Connection Establishment Performance of a Carrier Grade NAT Device. AINTEC '14. pp. 1-8. Nov. 2014.

国内会議 (査読有り)

- 明石邦夫, 井上朋哉, 安田真悟, 村本衛一, 知念賢一, 宇多仁, 篠田陽一. リンクエミュレータの多重実行の限界値測定. Internet Conference 2009, pp. 33—39, Oct 2009.
- 安田真悟，宮地利幸，ラズバン・ベウラン，牧野義樹，明石邦夫，井上朋哉，三輪信介，宇多仁，丹康雄，篠田陽一「空間情報に基づく災害時シナリオを用いたネットワークエミュレーション基盤の構築」，CSISDAYS2012 研究アブストラクト集, pp.61 2012-11
- 村本 衛一， 香林 誠， 石井 秀教， 明石 邦夫， 知念 賢一， 篠田 陽一，“無線網に対応した実時間映像音声伝送制御の検証のための時間推移ネットワークエミュレーション”，インターネットコンファレンス 2014 論文集，日本ソフトウェ ア科学会，研究会 資料シリーズ No.74, ISSN 1341-870X, pp.51-59, 2014 年 10 月

国内研究会

- 明石邦夫, 井上朋哉, 篠田陽一. インターネットの特性を考慮した実験ネットワーク構築フレームワークの設計と実装. 信学技報, vol. 113, no. 140, IN2013-37, pp. 7-12, 2013 年 7 月.
- 田部英樹, 明石邦夫, 宇多仁, 篠田陽一, 三輪伸介. データセンタネットワークの技術動向と相互接続性. IA2011, pp. 43-47. Jan 2012
- 高野 祐輝, 三浦 良介, 安田 真悟, 明石 邦夫, 井上 朋哉. SF-TAP: 柔軟で規模追従可能なトラフィック解析基盤の設計. 信学技報, vol. 114, no. 389, ICM2014-33, pp. 7-12, 2015 年 1 月.

その他

- 明石邦夫. 大規模ネットワークの実証環境向け抽象化技術に関する研究, 2010/03, 修士論文

# 参考文献

[1] Mark Weiser. Human-computer interaction. chapter The Computer for the 21st Century, pp. 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[2] Kevin Ashton. That internet of things thing. *RFiD Journal*, 2002.

[3] Friedemann Mattern and Christian Floerkemeier. *From the Internet of Computers to the Internet of Things*, Vol. 6462 of *LNCS*, pp. 242–259. Springer, 2010.

[4] Lora alliance. `https://www.lora-alliance.org/`.

[5] Juan-Carlos Ziga and Benot PONSARD. SIGFOX System Description. Internet-Draft draft-zuniga-lpwan-sigfox-system-description-01, Internet Engineering Task Force, October 2016. Work in Progress.

[6] Daniel Mahrenholz and Svilen Ivanov. Real-time network emulation with ns-2. *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp. 29–36, 2004.

[7] Technologies. Scalable Networks. QualNet. http://www.scalable-networks.com/.

[8] OPNET Technologies. Opnet network simulator. http://www.opnet.com/.

[9] Shilpa Bade, Meeta Kumar, and Pooja Kamat. A reactive energy-alert algorithm for manet and its impact on node energy consumption. *International Journal of Computer Applications*, Vol. 71, No. 18, pp. 1–6, June 2013. Full text available.

[10] Hoo-Rock Lee, Kyung-Yul Chung, and Kyoung-Son Jhang. A study of wireless

sensor network routing protocols for maintenance access hatch condition surveillance,. *Journal of Information Processing Systems*, Vol. 9, No. 2, pp. 237–246, 2013.

[11] Aditi Kumari, Neha Gandotra, S. C. Gupta, and Shrikant Upadhyay. Performance analysis of aodv, dsr, olsr and dsdv routing protocols using ns2 simulator. *Procedia Engineering*, Vol. 30, pp. 69–76, 2012.

[12] Jouni Malinen. mac8011_hwsim. `http://wireless.kernel.org/en/users/Drivers/mac80211_hwsim`, 2008.

[13] Bluez. http://www.bluez.org/.

[14] Linux ieee 802.15.4 implementation. https://www.kernel.org/doc/Documentation/networking

[15] Joakim Eriksson, Fredrik Osterlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, and Thiemo Voigt. Cooja/mspsim: Interoperability testing for wireless sensor networks. *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, pp. 27:1–27:7, 2009.

[16] Mohammad-Mahdi Moazzami. Orbit: A smartphone-based system platform for embedded sensing applications. Technical Report MSU-CSE-13-11, Department of Computer Science, Michigan State University, East Lansing, Michigan, December 2013.

[17] Katerina Pechlivanidou, Kostas Katsalis, Ioannis Igoumenos, Dimitrios Katsaros, Thanasis Korakis, and Leandros Tassiulas. Nitos testbed: A cloud based wireless experimentation facility. *IEEE 26th International Teletraffic Congress*, pp. 1–6, September 2014.

[18] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, Vol. 38, No. 2, pp. 69–74, March 2008.

[19] Pramodsanaga, JonathonDuering, RobertRicci, and Jay Lepreau. Modeling and

emulation of internet paths. *Sixth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 199–212, Apr 2009.

[20] Ramon R Fontes, Samira Afzal, Samuel HB Brito, Mateus AS Santos, and Christian Esteve Rothenberg. Mininet-wifi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pp. 384–389. IEEE, 2015.

[21] Shingo Yasuda, Kunio Akashi andToshiyuki Miyachi, Razvan Beuran, Yoshiki Makino, Tomoya Inoue, Shinsuke Miwa, and Yoichi Shinoda. Emulation-based ict system resiliency verification for disaster situations. *International Workshop on Resilient Internet based Systems (REIS 2013) in SITIS 2013*, pp. 875–882, December 2013.

[22] Razvan Beuran, Shingo Yasuda, Tomoya Inoue, Shinsuke Miwa, and Yoichi Shinoda. Using emulation to validate post-disaster network recovery solutions. *The 7th International ICST Conference on Simulation Tools and Techniques*, pp. 92–97, March 2014.

[23] Brian Sipos, Michael Demmer, Joerg Ott, and Simon Perreault. Delay-Tolerant Networking TCP Convergence Layer Protocol Version 4 . Internet-Draft draft-ietf-dtn-tcpclv4-01, Internet Engineering Task Force, November 2016. Work in Progress.

[24] Open-Mesh.net. B.a.t.m.a.n. (better approach to mobile ad-hoc networking). `http://www.open-mesh.net/`.

[25] Optimized link state routing protocol (olsr), 2003.

[26] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication*, Vol. 33, pp. 2–13, July 2003.

[27] Toshiyuki Miyachi, Takeshi Nakagawa, Ken-ichi Chinen, Shinsuke Miwa, and

Yoichi Shinoda. StarBED and SpringOS Architectures and Their Performance. In Thanasis Korakis, Hongbin Li, Phuoc Tran-Gia, and Hong-Shik Park, editors, *Testbeds and Research Infrastructure. Development of Networks and Communities - 7th International ICST Conference,TridentCom 2011, Shanghai, China, April 17-19, 2011, Revised Selected Papers*, Vol. 90 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 43–58. Springer, 2011.

[28] Starbed. http://starbed.nict.go.jp/.

[29] Razvan Beuran, Lan Tien Nguyen, Khin Thida Latt, Junya Nakata, and Yoichi Shinoda. Qomet: A versatile wlan emulator. *IEEE International Conference on Advanced Information Networking and Applications (AINA-07)*, pp. 21 – 23, 2007.

[30] Priya Mahadevan, Adolfo Rodriguez, David Becker, and Amin Vahdat. Mobinet: A scalable emulation infrastructure for ad hoc and wireless networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, Vol. 10, No. 2, pp. 26–37, April 2006.

[31] Harald Welte and Pablo Neira Ayuso. libnetfilter_queue project. `http://www.netfilter.org/projects/libnetfilter_queue/index.html`, June 2008.

[32] A Ihara, S Murase, and Kunio Goto. Ipv4/v6 network emulator using divert socket. *18th International Conference on Systems Engineering(ICSE2006)*, pp. 159 – 166, 2006.

[33] Yusuke Sugiyama and Kunio Goto. Design and implementation of a network emulator using virtual network stack. The 7th International Symposium on Operations Research and Its Applications (ISORA' 08), pp. 351 – 358, Nov 2008.

[34] S Hemminger. Network emulation with netem, April 2005.

[35] Mark Carson and Darrin Santay. Nist net - a linux-based network emulation tool. *ACM SIG- COMM Computer Communications Review*, Vol. 33, No. 3, pp. 111–126, July 2003.

[36] Linux advanced routing & traffic control. `http://lartc.org/`, 2002.

[37] LuigiRizzo. Dummynet:asimpleapproachtothe evaluation of network protocols. *ACM SIGCOMM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, Jan 1997.

[38] libnl Project. netlink library. `http://people.suug.ch/~tgr/libnl/`.

[39] Kunio Akashi. Meteor. `https://github.com/k-akashi/meteor`, 2011.

[40] open80211s. `http://open80211s.org/open80211s/`.

[41] Jun Jangeun, Pushkin Peddabachagari, and Mihail Sichitiu. Theoretical maximum throughput of IEEE 802.11 and its applications. In *Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003.*, pp. 249–256. IEEE Comput. Soc, 2003.

[42] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348 (Informational), August 2014.

[43] Yasuhiro Ohara, Kaname Nishizuka, Ken'ichi Chinen, Kunio Akashi, Makoto Kohrin, Eiichi Muramoto, and Shin Miyakawa. On the impact of mobile network delays on connection establishment performance of a carrier grade nat device. In *Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference*, AINTEC '14, pp. 1:1–1:8, New York, NY, USA, 2014. ACM.

[44] G. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke. Ieee 802.11s: The wlan mesh standard. *IEEE Wireless Communications*, pp. 104–111, Feb 2010.

[45] S. Perreault, I. Yamagata, S. Miyakawa, A. Nakagawa, and H. Ashida. Common Requirements for Carrier-Grade NATs (CGNs). RFC 6888 (Best Current Practice), April 2013.