無線ネットワークTAPデバイスとネットワークシミュレータを連携した無線LANエミュレータの実現

加藤 新良太[1]　高井 峰生[2,3]　石原 進[4]

概要：無線 LAN エミュレーションは，実際のアプリケーションやオペレーティングシステム（OS）とネットワークシミュレータを組み合わせることで，アプリケーションやネットワークプロトコルスタックの動作はそのままに，変調や送信電力制御等の無線 LAN デバイスの動作や，電波伝搬，無線通信端末のモビリティ等の物理的な挙動部分をシミュレータで再現する技術であり，システム開発・評価に有用である．従来の無線 LAN エミュレーション技術では，無線 LAN を介したデータ通信は模擬できるものの，OS による送信電力や通信周波数等の無線 LAN の通信制御に関わる情報をシミュレータと OS・アプリケーション間でやりとりできないため，送信電力の調整や送信周波数の変更等の無線 LAN のデバイス操作を模擬できない．そこで，著者らは，実動する OS・アプリケーションを用いた無線 LAN でのデータ通信とデバイス操作の双方を模擬できる無線 LAN エミュレーションフレームワークを提案している．本稿では，エミュレーションフレーワークに既存のネットワークシミュレータを組み込む際に課題となるネットワークシミュレータのフレームワーク内での配置方法と，実際の OS・アプリケーションとシミュレータ間で情報を交換する方法の検討と設計を行った．この結果，模擬するネットワークの規模が大きくない場合には，同一マシン上でシミュレータと実際の OS・アプリケーションを配置し，既存のエミュレーション技術でデータ通信を模擬しつつ，無線 LAN エミュレーションフレームワークで無線 LAN のデバイス操作を模擬する方法がエミュレーションのリアルタイム性とその結果の正確さの観点から適当であることを示した．

Realization of a Wireless LAN Emulator using a Wireless Network Tap Device and a Network Simulator

Arata Kato[1]　Mineo Takai[2,3]　Susumu Ishihara[4]

1. はじめに

ネットワークプロトコルやネットワークアプリケーションの動作検証及び性能評価の方法の 1 つにネットワークエミュレーションがある．ネットワークエミュレーションは，オペレーティングシステム（OS）で実際に動作するアプリケーションやネットワークプロトコルスタックをネットワークシミュレータと接続し，それらのソフトウェアが送信・受信するデータパケットをネットワークシミュレータに転送する．これにより，ネットワークデバイスの動作，ネットワークノード間の転送遅延，及びパケットロス等のネットワーク性能に影響する挙動をシミュレーションモデルに基づいて計算し，実フレームの送信前に遅延の挿入や帯域を制限することで，実機で通信する際のアプリケーションやネットワークプロトコルの挙動を再現できる．特に，車車間通信のような，実験による動作検証が困難であり，かつ通信ノードのモビリティが高く，電波伝搬特性の変化が無視できない場合における OS・アプリケーションの動作検証に有用である．

筆者らは [1] 及び [2] において，無線 LAN を対象としたネットワークエミュレーションを実現する無線ネットワー

[1] 静岡大学大学院総合科学技術研究科
Graduate School of Integrated Science and Technology, Shizuoka University.
[2] Computer Science Department, University of California, Los Angeles.
[3] 大阪大学大学院情報科学研究科
Graduate School of Information and Technology, Osaka University.
[4] 静岡大学学術院工学領域
College of Engineering, Shizuoka University.

ク TAP デバイスと，それを用いた無線 LAN エミュレーションフレームワークを提案した．筆者らのエミュレーションフレームワークでは，MAC 副層の CSMA/CA と物理層の電波伝搬や通信ノードのモビリティ等の挙動をネットワークシミュレータで模擬するとしていた．しかしながら，既存のネットワークシミュレータと筆者らの無線 LAN エミュレーションフレームワークを組み合わせる場合，フレームワーク内におけるネットワークシミュレータの配置方法とシミュレータと実際の OS・アプリケーション間の情報交換の方法を決める必要がある．

ネットワークシミュレータと連携させたネットワークエミュレーションの実現方法は，用途や実装によって様々であるが，ns-3[3]，Scenargie[4] 等のネットワークシミュレータは，ネットワーク TAP デバイスと呼ばれる仮想有線 LAN デバイスを用いて OS とネットワークシミュレータ間を接続する．ネットワーク TAP デバイスは，OS に対して Ethernet デバイスとして振る舞い，あるユーザプロセスから送信されたデータパケットを OS の IP プロトコルスタックから受け取ると，そのデータパケットを別のユーザプロセスにファイルディスクリプタを介して直接転送する．このため，ユーザプロセスから見て，実際は OS のメモリ空間を介して交換されるパケットを，Ethernet ケーブルを介して交換されているように見せることができる．

しかしながら，無線 LAN による通信では，データ転送処理の他に，送信電力や通信周波数の設定，受信信号強度の把握等の無線 LAN デバイスの制御操作や，通信ノード間で BSSID (Basic Service Set Identifier) 等の無線 LAN の通信制御に必要な情報の交換が伴う．無線 LAN での通信を正確に模擬するためには，データ転送，通信制御の 2 つを同時に模擬できることが求められる．このため，無線 LAN での通信を正確に模擬するためには，**図 1** に示すように，実動する OS・アプリケーションによる通信制御も模擬することが求められる．

筆者らは，文献 [1] において，無線 LAN フレームによるデータ転送と本物の無線 LAN アプリケーションや OS によって設定された無線 LAN の制御情報の双方をシミュレータに転送できる無線 LAN に対応した無線ネットワーク TAP デバイスを設計・実装した．無線ネットワーク TAP デバイスは，Linux 向けに実装された仮想無線 LAN デバイスであり，Linux の無線 LAN 制御システムに対して自身が無線 LAN デバイスとして振舞うことで，無線 LAN フレームと，無線 LAN の通信制御に用いられる情報の双方を任意のユーザプロセスとの間で交換できる．

また，文献 [2] において，無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワークの設計及びそのプロトタイプを実装し，無線 LAN フレーム及び無線 LAN の通信制御情報の双方をユーザプロセスと交換できることを示した．一方で，MAC 副層の CSMA/CA

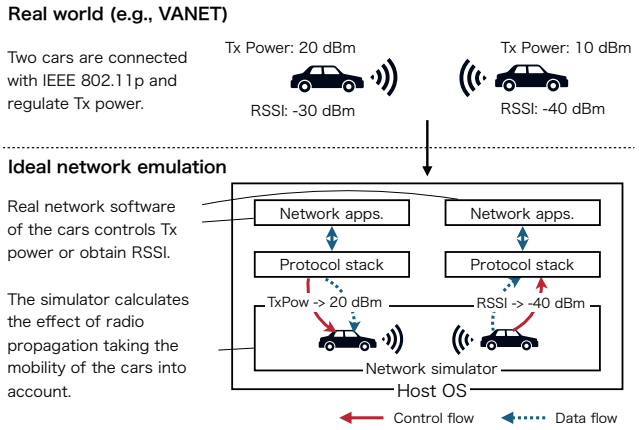


図 1　筆者らが目指す無線 LAN エミュレーション
Fig. 1　WLAN emulation we aim to realize.

の動作や，物理層における電波伝搬や通信ノードのモビリティ等の挙動を再現する方法の検討が十分ではなかった．

そこで，本稿では，既存のネットワークシミュレータを筆者らのエミュレーションフレームワークに組み込むことで，CSMA/CA 及び物理層の挙動を含めた無線 LAN エミュレーションの実現を目指し，既存のシミュレータをフレームワーク内に組み込む際に課題となるフレームワーク内におけるシミュレータ機能の配置方法と，シミュレータとネットワークアプリケーション間での情報交換の方法について議論する．具体的には，ネットワークアプリケーションとシミュレータ機能を同一ホストに配置する方法と，シミュレータ機能を別のマシンに配置する方法を検討する．また，情報交換の方法として，無線 LAN フレームと制御情報の双方をユーザ空間を介してシミュレータに転送する方法と，無線 LAN フレームをカーネル空間から転送しつつ，制御情報をユーザ空間を介して別々に転送する方法について設計する．

以下，2 章にて既存のネットワークシミュレータが実装するエミュレーション機能と，シミュレータ及び実際の OS・アプリケーション間の接続方法の違いについて述べ，3 章で無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワークの概要と，既存のネットワークシミュレータを組み込む際に生じる課題について述べる．4 章にて，フレームワーク内のシミュレータ機能の配置方法と，その実際の OS・アプリケーションとネットワークシミュレータ間の接続方法を検討し，5 章でそれらの考察を述べ，6 章で本論文のまとめを述べる．

2. 関連研究

既存のネットワークシミュレータでは，ネットワーク TAP デバイスを用いることで，ネットワークプロトコルスタックやアプリケーション等の OS で実動するソフトウェアとネットワークシミュレータを接続している．一方で，ns-3[3] や Mininet[7] 等の一部のネットワークシミュレー

タが実装するエミュレーション機能では，ネットワークTAP デバイス以外の方法により実動するソフトウェアとシミュレータ間を接続するものがある．以下，既存のネットワークエミュレーション技術で用いられている実際のソフトウェアとネットワークシミュレータ間の接続に用いられるインタフェースとその実装方法に着目し，それらの特徴ついて述べる．

2.1 Scenargie Emulation Module

Scenargie Emulation Module は，ネットワークシミュレータ Scenargie[4] 上で，Linux のアプリケーションや IP スタックを実行可能とする拡張機能である．このモジュールでは，ネットワーク TAP デバイスを用いてネットワークシミュレータと IP スタックを接続し，実際の IP パケットを用いてシミュレータがデータリンク層及び物理層の挙動を模擬する．しかし，無線 LAN によるデータ通信を模擬することを考えた場合，ネットワーク TAP デバイスでは Linux の無線 LAN 制御システム（Linux Wireless Subsystem）と無線 LAN デバイス間でやりとりされる送信電力，通信周波数及び受信信号強度等の無線 LAN デバイスの挙動や受信状況の把握に必要な情報をやりとりできないため，Linux による無線 LAN の通信制御の動作をエミュレーション評価に反映させることが難しい．

2.2 ns-3 (Direct Code Execution)

Direct Code Execution (DCE) は，Linux カーネルのネットワークプロトコルスタックを ns-3 上で実行できる拡張機能である [5]．DCE では，Linux カーネルで定義されるネットワークインタフェース（`net_device` 構造体）を用いて ns-3 と Linux カーネルのネットワークプロトコルスタックを接続する．`net_device` 構造体は，ユーザプロセスに対して Linux システムに接続されているネットワークデバイスを有線 LAN デバイス及び無線 LAN デバイスを含め抽象化し，ユーザプロセスに対してファイルシステムのシステムコールと同様にネットワークデバイスにアクセス可能とする API (Application Programming Interface) を提供する．

ns-3 は，内部に `net_device` 構造体と類似した仮想ネットワークインタフェース (ns3::NetDevice) を内部に定義しており，DCE は，Linux カーネルの `net_device` 構造体と ns-3 内部の ns3::NetDevice に格納されているデバイス情報を同期させることで，ns-3 が Linux カーネルによる `net_device` 構造体にあるデバイス情報の操作を把握できる仕組みを提供する．しかし，`net_device` 構造体は，Linux システムに接続されている有線 LAN 及び無線 LAN デバイスを統合化するインタフェースであるため，OS やアプリケーションが変調や送信電力制御等の無線 LAN の通信制御を行っても，ns-3 にはそれらの制御情報が通知されないため，無線 LAN での通信制御を完全には模擬できない．

2.3 Mininet-WiFi

Fontes らは，ネットワークシミュレータ Mininet[7] 上で，Linux で実際に動作する無線 LAN アプリケーションを実行可能なエミュレーションモジュール Mininet-WiFi を提案している [8]．Mininet-WiFi では，Linux の仮想 Ethernet デバイス (`veth` デバイス)[9] と Open vSwitch[10] を用いてネットワークシミュレータとアプリケーションを接続する．Open vSwitch は，ネットワーク層及びデータリンク層で動作する仮想ネットワークスイッチである．`veth` デバイスは，ネットワーク TAP デバイス同様に Linux で実装されている仮想 Ethernet デバイスの 1 つであるが，ネットワーク TAP デバイスは，OS のネットワークプロトコルスタックから受け取った Ethernet フレームをファイルシステムを介してアプリケーションに転送するのに対し，`veth` デバイスは，2 つの `veth` デバイスで 1 つのペアを形成し，同じペアの `veth` デバイス間で Netlink ソケットを用いて Ethernet フレームをやりとりする．

当然ながら，Ethernet フレームのヘッダには無線 LAN の通信制御に関する情報は含まれていないため，Mininet-WiFi は，OS による無線 LAN デバイスの制御操作や無線 LAN の通信制御を模擬できない．なお，Mininet-WiFi では，アプリケーションと `veth` 間のインタフェースにはソケット API が用いられる．

2.4 VirtualMesh

Staub らは，ネットワークシミュレータ OMNeT++[11] と，Linux のネットワークアプリケーションを接続するエミュレーションフレームワーク VirtualMesh を提案している [12]．VirtualMesh は，ネットワークシミュレータ OMNeT++と Linux のネットワークアプリケーション間の接続に PacketModeller と呼ばれる独自のアプリケーションを用いている．

PacketModeller は，ネットワーク TAP デバイスを用いて取得した Ethernet フレームを，仮想の無線 LAN デバイスの情報を含めた独自定義のメッセージにカプセル化し，そのメッセージを再度 Ethernet フレームにカプセル化してネットワークシミュレータに転送する．このため，1 つの Ethernet フレームをネットワークシミュレータに転送する際に，独自フォーマットのメッセージのカプセル化/非カプセル化の際にオーバヘッドが生じる他，1 つの無線 LAN フレームを複数の Ethernet フレームに分割してネットワークシミュレータに転送するため効率が悪い．

2.5 既存のネットワークエミュレーション技術における実装上の特徴

表 1 に示すように，既存のネットワークシミュレータで実装されているエミュレーション機能において，シミュレータと実際の OS・アプリケーション間の接続に用いら

表 1　エミュレーション機能を有するネットワークシミュレータと，OS-シミュレータ間のインタフェース
Table 1　Network simulators with an emulation function and these interfaces

シミュレータ	インタフェース
Scenargie[4]	ネットワーク TAP デバイス
ns-3[3]	`struct net_device`（`bridge` インタフェース）
Mininet-WiFi[7]	`veth` デバイス & Open vSwitch
VirtualMesh[12]	PacketModeller (独自アプリケーション)

れているインタフェースは，ネットワーク TAP デバイス，`veth` デバイス，仮想ネットワークインタフェース (`struct net_device`，`bridge` インタフェース)，もしくは独自のアプリケーションを用いることで，実機で動作するソフトウェアを接続している．しかしながら，いずれの方法においても共通している点として，ネットワークシミュレータと実際の OS・アプリケーション間で Ethernet フレームをやりとりすることが挙げられ，無線 LAN フレームや，無線 LAN の通信制御に用いられる情報のやりとりについて考慮されていない．

3. 無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワーク

本章では，本稿においてネットワークシミュレータと連携させる無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワークの概要及びその動作について述べ，そのエミュレーションフレームワークとネットワークシミュレータを連携させた場合に生じる課題について述べる．

3.1 無線 LAN エミュレーションフレームワークの概要

本稿で述べる無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワークは，従来のネットワークエミュレーションでは実現されていなかった実際の無線 LAN フレームや，送信電力及び受信信号強度等の無線 LAN フレームの送信制御や受信状況の把握に関する情報を用いたエミュレーションを可能とする．具体的には，Linux の無線 LAN 制御システム（Linux Wireless Subsystem）と無線 LAN デバイスドライバ間でやりとりされる実際の無線 LAN フレームに加えて，本物の無線 LAN フレームと無線 LAN の制御情報を無線ネットワーク TAP デバイスを経由してネットワークシミュレータに転送する．これにより，ネットワークシミュレータが無線 LAN で通信する際の実際の OS・アプリケーションの挙動を把握できる．

3.2 無線ネットワーク TAP デバイス

無線ネットワーク TAP デバイス（wtap80211）は，著者らが設計・実装した Linux で動作する仮想無線 LAN デバイスである [1]．この仮想デバイスは，**図 2** に示すように Linux の仮想無線 LAN ドライバとして実装されており，MAC アドレス，対応周波数，対応無線規格等から成る仮想の無線 LAN デバイスの構成情報を Linux の無線 LAN 制御システム (Linux Wireless Subsystem) に登録し，Linux システムに本物の無線 LAN デバイスが接続されているように振る舞う．

表 2　wtap80211 の仮想無線 LAN デバイスの構成
Table 2　Property of a virtual WLAN device of wtap80211.

OS	Linux 3.19.x
対応規格	IEEE 802.11a/b/g/n/ac, IEEE 802.11p/s/ad（一部に対応）
動作モード	Managed/Master/IBSS (Ad-hoc)/ Monitor/Mesh/OCB

表 2 に無線ネットワーク TAP デバイスが Linux の無線 LAN 制御システムに登録する仮想デバイスの構成例を示す．無線ネットワーク TAP デバイスは，一般用途向けの IEEE 802.11n/ac 等の無線 LAN 規格に対応する仮想無線 LAN デバイスを作成することが可能である．また，Linux システムにインストールされている無線 LAN ツールのバージョンに依存するものの，車車間・路車間通信向けの無線 LAN 規格 IEEE 802.11p や，ミリ波向け無線 LAN 規格 IEEE 802.11ad に対応した仮想無線 LAN デバイスを作成することもできる．

無線ネットワーク TAP デバイスは，Linux カーネル内において仮想の無線 LAN デバイスドライバとして実装されており，Linux の無線 LAN 制御システムと無線 LAN デバイスドライバ間で取り交わされるフレームデータや，BSSID やチャネル等の通信制御に用いられる情報を全てキャプチャできる．また，Netlink ソケットを通じてユーザプロセスとそれらのデータを直接交換できる．このため，QoS 制御の際に用いられる無線 LAN デバイスのファームウェアにあるメッセージキューの操作や，ネットワークアプリケーションがクロスレイヤ制御をした際の挙動をユーザプロセスが把握できるため，実際の OS・アプリケーションによる詳細な通信制御に基づいたエミュレーションが可能となる．

3.3 wtap80211d (API)

wtap80211d は，無線ネットワーク TAP デバイスとネットワークシミュレータ間のインタフェースとして動作するデーモンアプリケーションである．このデーモンアプリケーションは，無線ネットワーク TAP デバイスとの間に Netlink ソケットで接続されており，無線ネットワーク TAP デバイスから無線 LAN フレームや通信制御の情報が格納された Netlink メッセージを交換する．ネットワーク

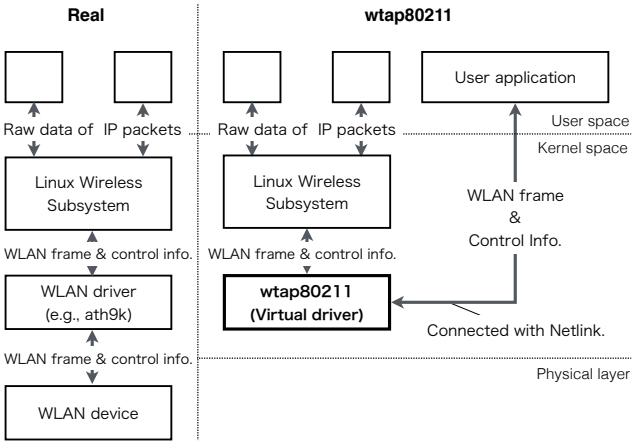


図 2　wtap80211 のアーキテクチャ
Fig. 2　Architecture of a Wireless Network Tap Device

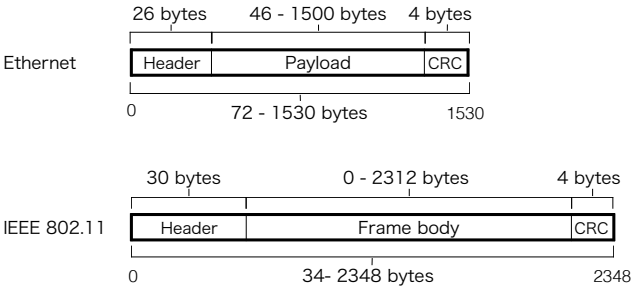


図 3　Ethernet と IEEE 802.11 のフレームフォーマットの比較
Fig. 3　Comparison of frame formats of Ethernet and IEEE 802.11.

アプリケーションとの間では Linux カーネルの SocketAPI を介してデータを交換する．

3.4　エミュレーションフレームワークに既存のネットワークシミュレータを組み込む際の課題

既存のネットワークシミュレータを筆者らのフレームワークに組み込む際の課題として，無線 LAN フレームや制御情報を実際の OS・アプリケーションから既存のネットワークシミュレータに転送する際に，これらのデータを 1 つの Ethernet フレームに格納できないことが挙げられる．既存のネットワークエミュレーション技術では，IP パケットもしくは Ethernet フレームをネットワークシミュレータに転送する設計となっている．ところが，**図 3** に示した Ethernet 及び IEEE 802.11 のフレームフォーマットを比較してわかるように，無線 LAN フレームの長さはヘッダ部分も含めると最大 2348 bytes である一方で，Ethernet フレームのペイロード長は最大 1500 bytes である．このため，1 つの無線 LAN フレームをネットワークシミュレータに転送するためには，無線 LAN フレームを複数の Ethernet フレームに分割して送信する必要がある．この際に，分割された無線 LAN フレームを元の 1 つの無線 LAN フレームに再構築する処理が伴うため，無線 LAN フレームの転送時にオーバヘッドが生じる．無線 LAN フレームの送信・受信の際には，無線 LAN フレームのほかに，フレーム毎に送信電力や通信周波数，及び受信信号強度等の送信制御及び受信状態の把握に用いられる情報が付随する．このため，無線 LAN フレームとそれらの制御情報をネットワークシミュレータに転送する場合，無線 LAN フレームを転送する Ethernet フレームと，制御情報を転送するための Ethernet フレームを合わせて 2 つ以上の Ethernet フレームをやりとりする必要がある．このため，無線 LAN フレーム及び通信制御情報の双方を転送可能であり，かつ無線 LAN フレームを効率的に転送する仕組みが求められる．

4.　無線 LAN エミュレーションフレームワークとネットワークシミュレータを繋ぐインタフェースの設計

本章では，筆者らが実現を目指す無線 LAN エミュレータの実装にあたり，3 章にて述べた無線 LAN エミュレーションフレームワークと既存のネットワークシミュレータを接続する際に生じる課題に対する解決方法について議論する．具体的には，シミュレータ機能の物理的な配置方法と，シミュレータ及び実際の OS・アプリケーション間の接続方法の 2 つを考慮した解決方法を検討する．

4.1　設計目標

シミュレータ機能の配置方法と，シミュレータと実際の OS・アプリケーション間の接続方法の設計にあたり，共通して考慮すべき課題として，(1) リアルタイム性の確保と，(2) シミュレータに加える変更を最小化することが挙げられる．

4.1.1　リアルタイム性の確保

エミュレーション環境上で動作する実際のアプリケーションやネットワークプロトコルスタックは実時間で動作するため，それらのソフトウェアを実機動作時の振る舞いをエミュレーション環境上で模擬するためには，ネットワークシミュレータと連携させた場合にリアルタイム性の確保が必要である．このためには，エミュレーションフレームワークとネットワークシミュレータ間で，無線 LAN フレームや無線 LAN の制御情報を転送する際のオーバヘッドを可能な限り小さくする必要がある．車車間・路車間通信等の通信ノードのモビリティが高い場合，転送遅延にわずかな誤差が生じるとシミュレーション上の電波伝搬特性が大きく変わってしまうため，正確なエミュレーション結果を得るためには，ネットワークシミュレータと実際の OS・アプリケーション間の転送遅延は可能な限り小さくする必要がある．

4.1.2　シミュレータ側の改変部分の最小化

ns-3 や Scenargie 等の既存のネットワークシミュレータには，既に無線 LAN のエミュレーション評価のために，電波伝搬モデルや通信ノードのモビリティモデルなど，多

種多様な拡張モデルが実装されている．それらを筆者らのエミュレーションフレームワークで活用するためには，その拡張モジュールが動作するように，ネットワークシミュレータに加える変更を最小限に止めることが必要である．

4.2 シミュレータ機能の配置方法

フレームワーク内におけるシミュレータ機能の配置方法には，その利用用途によって，(1) ネットワークアプリケーションとシミュレータを同一ホスト上に配置する方法と，(2) シミュレータを別のマシンに配置する方法の 2 つの配置方法が考えられる．

4.2.1 シミュレータ機能と実際の OS・アプリケーションを同一マシンに配置する方法

1 つ目の方法は，**図 4(a)** に示すように，実際の OS・アプリケーションと既存のネットワークシミュレータを，同じ計算機上で全て実行する．この方法では，エミュレーション環境の構築に 1 台の計算機があれば良いため，環境構築が容易であることが利点として挙げられる．しかし欠点として，模擬したいネットワークの規模が大きく，想定される無線通信ノードの数が多い場合，シミュレーションやエミュレーション環境上で動作するアプリケーションの処理負荷が高くなることで処理遅延が増大し，リアルタイム性が損なわれる恐れがある．

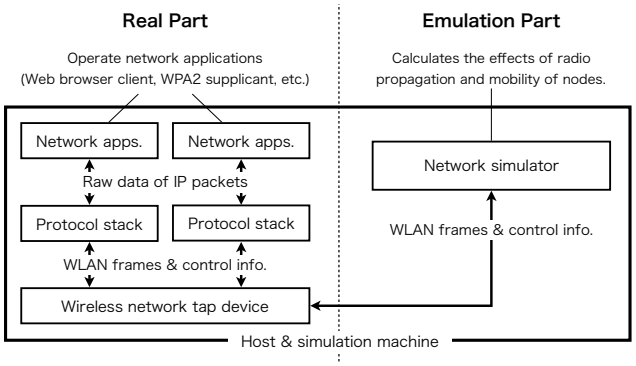
4.2.2 シミュレータ機能を別マシンに配置する方法

2 つ目の方法は，**図 4(b)** に示すように，ネットワークシミュレータと実際の OS・アプリケーションをそれぞれ Ethernet で接続された別々のマシンで実行する．この方法の利点は，エミュレーションの処理負荷を分散し，同一ホストでエミュレーションをした場合に比べ，より多数の通信ノードを同時に模擬できる点である．また，この方法の応用として，**図 4(c)** に示すように，実機に無線 LAN エミュレーションフレームワークを構築し，実機とネットワークシミュレータを接続することで，実際に実環境で使われているハードウェアとソフトウェアを用いたエミュレーション評価もできると考えられる．

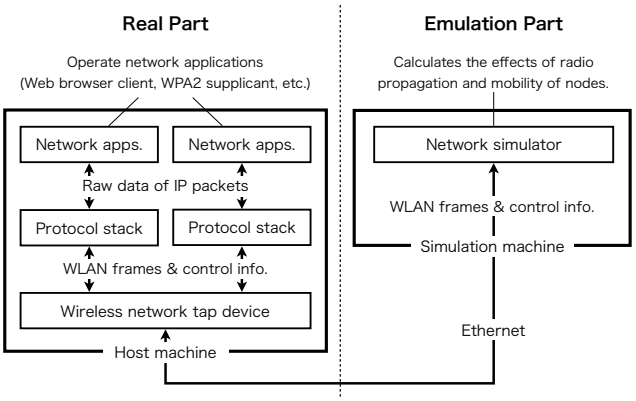
しかしながら，シミュレータを実行するマシンと，フレームワークを実行するマシンを別々にした場合，無線 LAN フレームやその制御情報等を Ethernet フレームにカプセル化して転送する必要がある．この際に，無線 LAN フレームを複数の Ethernet フレームに分割して送信・受信しなければならないため，そのカプセル化/非カプセル化の処理遅延がオーバヘッドとなる．

4.3 無線ネットワーク TAP デバイスとシミュレータ間の情報交換方法の設計

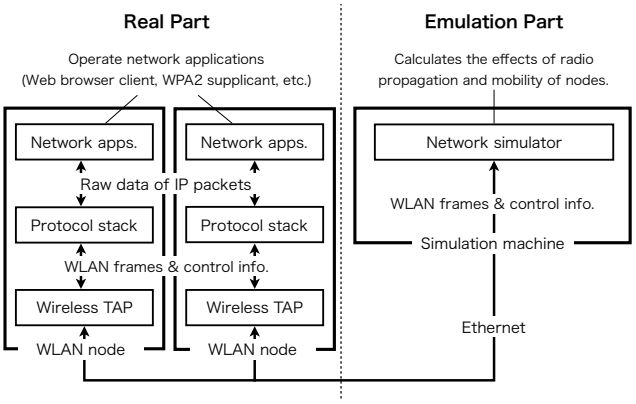
本節では，4.1 節で述べた設計目標を満たすシミュレータ機能と実際の OS・アプリケーション間で無線 LAN フレームや制御情報を交換する方法として，(1) wtap80211d (API) のみを用いて接続する方法，(2) wtap80211d (API) とネットワーク TAP デバイスを併用する方法，(3) netpoll ドライバを用いる方法の計 3 つの方法について述べる．



(a) シミュレータ機能と実際の OS・アプリケーションを同一マシンに配置する方法



(b) シミュレータ機能を別のマシンに配置する方法



(c) 実機とシミュレータ機能を別々に配置する方法

図 4 エミュレーション環境の物理構成例

Fig. 4 Examples of physical configuration of an emulation environment.

4.3.1 wtap80211d (API) を用いて接続する方法

1 つ目は，**図 5(a)** に示すように，無線ネットワーク TAP デバイスから無線 LAN フレームと無線 LAN の制御情報の双方を wtap80211d (API) 経由でネットワークシミュレータに転送する方法である．この方法では，アプリケーションがデータを送信すると，データパケットは Linux の無線 LAN 制御システムで無線 LAN フレームにカプセル化され，wtap80211d (API) に転送される．wtap80211d (API)

は，ネットワークシミュレータと Virtual Ethernet デバイス (veth デバイス) を用いてプロセス間通信を行い，無線ネットワーク TAP デバイスから転送された無線 LAN フレームを複数の Ethernet フレームに分割し，ネットワークシミュレータに転送する．

この方法では，無線ネットワーク TAP デバイスを通じて，実際の無線 LAN フレームや，送信電力及び受信信号強度等のフレームのやりとりに必要な制御情報全てをネットワークシミュレータに転送できるため，実際の OS・アプリケーションによる無線 LAN の通信制御の挙動をネットワークシミュレータが全て把握できる利点がある．

一方で，シミュレータで模擬する転送遅延やパケットロスの正確さが損なわれる可能性がある．無線 LAN フレームと制御情報は共通して wtap80211d (API) を経由して転送されるため，模擬端末数の増加や，無線 LAN フレームが連続して送信・受信されると，wtap80211 (API) に処理負荷が集中する．従って，エミュレーション環境上で動画や音声等の容量の大きいデータを交換する場合，ネットワークアプリケーション間でそれらのデータが正しくやりとりされない可能性がある．

4.3.2 wtap80211d (API) とネットワーク TAP デバイスを併用する方法

2 つ目は，**図 5(b)** に示すように，無線 LAN の制御情報を wtap80211d (API) を介してネットワークシミュレータに転送しつつ，無線 LAN フレームのペイロード部分をネットワーク TAP デバイスからネットワークシミュレータに転送する方法である．この方法の利点は，既存のネットワークシミュレータが実装している Ethernet フレームを用いたネットワークエミュレーション機能をそのまま実行できることである．一方で，無線ネットワーク TAP デバイスを通さずにデータがやりとりされるため，無線 LAN フレームのヘッダ部分をネットワークシミュレータに転送できないことが欠点として挙げられる．このため，フレームの送信・受信毎に付随する送信電力や通信周波数等の送信制御用のパラメータや，受信信号強度等の受信状態を把握に用いる情報をネットワークシミュレータと交換できない．

4.3.3 netpoll モジュールを用いる方法

3 つ目は，**図 5(c)** に示すように，無線ネットワーク TAP デバイスが，Linux カーネルに実装されてる netpoll モジュールを経由して，別マシンで稼働しているネットワークシミュレータに無線 LAN フレームや無線 LAN の制御情報を直接転送する方法である．netpoll モジュールは，予めカーネルパラメータとして設定された宛先/送信元 IP アドレスを元に，Linux のカーネルモジュールから転送されたデータを，擬似的に作成した UDP データグラムに格納する．その後，その UDP データグラムを IP パケットにカプセル化し，ネットワークデバイスを介して外部へ送信する．また，IP パケットの受信時は，IP パケットと UDP データグラムをそれぞれ非カプセル化したのち，UDP データグラムのペイロードにあるデータを Linux のカーネルモジュールへ転送する．

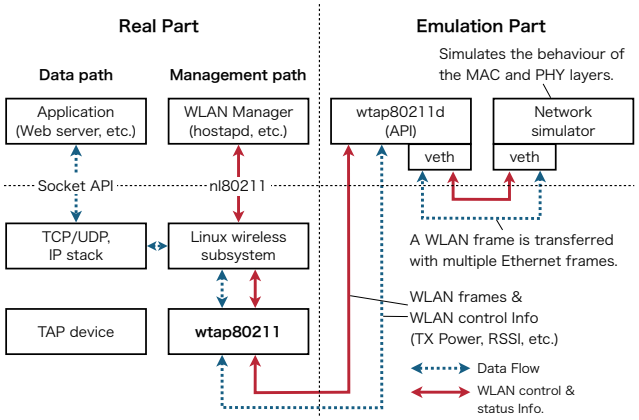
この方法は，無線ネットワーク TAP デバイスから直接ネットワークシミュレータに無線 LAN フレームや制御情報を転送でき，TCP/IP におけるルーティング，フラグメンテーション，再送処理等を行わないためオーバヘッドは小さい．しかしながら，実際の OS・アプリケーションとネットワークシミュレータ間におけるパケットロス等の通信障害に対応できないため，シミュレータが意図して模擬した転送遅延やパケットロスと，意図せずに生じた障害との区別が困難となり，エミュレーションの正確さを判定することが難しいことが欠点として挙げられる．

5. 考察

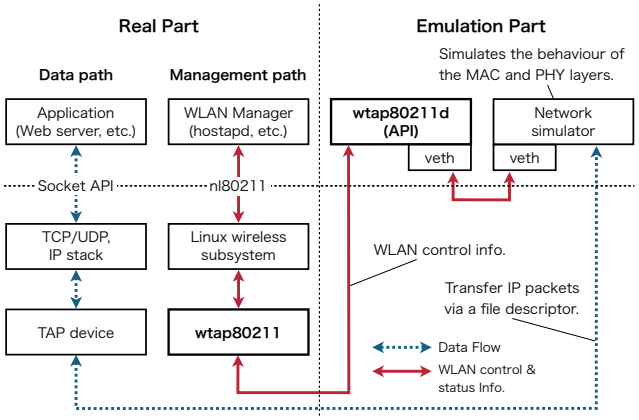
本稿で議論した無線 LAN エミュレーションフレームワークにおけるシミュレータ機能の配置方法と，ネットワークシミュレータと実際の OS・アプリケーション間の接続方法の特徴を，エミュレーションのリアルタイム性と規模性及び，エミュレーション結果の正確性についてまとめると，**表 3(a)** と**表 3(b)** に示す通りとなる．

ネットワークシミュレータと実際の OS・アプリケーションを実行する計算機を別々にすることで，処理負荷を分散でき，エミュレータの規模性を拡張することができると考えられる．一方で，その方法では，エミュレーションフレームワークとネットワークシミュレータ間で無線 LAN フレームや無線 LAN の通信制御に関する情報をやりとりする際に生じる転送遅延やパケットロスによるエミュレーション結果への影響を無視できない．このため，ネットワークアプリケーションからは，シミュレータによって意図された転送遅延・パケットロスなのか，物理マシン間で意図せずに生じたものなのか判断が難しい．よって，同一マシン上で実際の OS・アプリケーションとシミュレータを実行する方法が良いと考えられる．

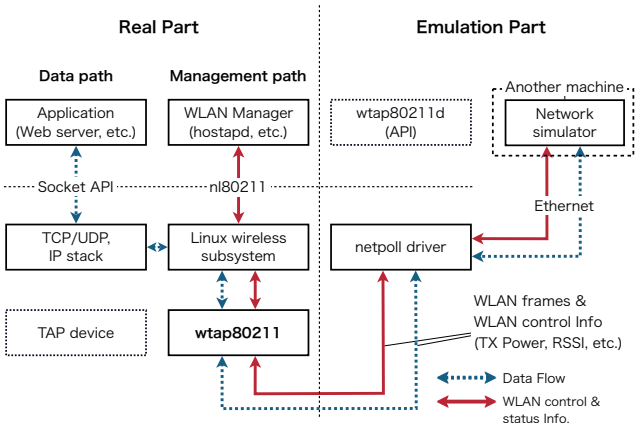
ただし，エミュレーションにおける規模性を重視してシミュレータ機能を別のマシンで実行する場合や，実機をシミュレータに接続する場合には，無線 LAN フレームやその制御情報を複数の Ethernet フレームに分割してシミュレータと実際の OS・アプリケーション間でやりとりする必要がある．そのような場合，ns-3 や Scenargie 等の既存のネットワークシミュレータでは，無線 LAN の MAC 副層及び物理層の挙動を模擬する際に，CSMA/CA 及び電波伝搬等の MAC 副層及び物理層における実機での処理遅延を考慮し，実環境への実フレーム送信前の転送遅延時間を計算している．したがって，無線 LAN フレームを複数の Ethernet フレームに分割して送信する場合，複数のフレーム転送にかかる処理遅延の合計が 1 つ無線 LAN フ



(a) wtap80211d (API) を用いて接続する方法



(b) wtap80211d (API) とネットワーク TAP デバイスを併用する方法



(c) netpoll モジュールを用いる方法

図 5　アプリケーションとシミュレータ機能の接続方法

Fig. 5　Methods for connecting a network simulator to network applications.

レームの転送遅延よりも短くなるように，シミュレータ側で Ethernet フレームの送信前に挿入する遅延を調整する必要がある．

既存のネットワークエミュレーションでは，OS とネットワークシミュレータ間のインタフェースでの処理遅延を考慮して，実環境へフレームを送信する前に模擬した遅延時間を挿入していると考えられるため，既に実装されているネットワークシミュレータのインタフェースを改変する場合，エミュレーション評価における転送遅延が正しく模擬されない可能性がある．このため，ネットワークシミュレータへの変更が最も少ない wtap80211d (API) とネットワーク TAP デバイスを併用する方法が，実際の OS・アプリケーションとシミュレータ間の接続方法として適している．

表 3　提案法の比較

Table 3　Comparison of proposed methods.

(a) シミュレータ機能の配置方法

配置方法	実時間性	規模性	正確性
A. 同一マシン	○	△	○
B. 複数マシン (実機無)	△	○	△
C. 複数マシン (実機有)	△	○	○

(b) 設計したシミュレータとアプリケーション間の接続方法

接続方法	実時間性	規模性	正確性	適合する配置方法
D. wtap80211d	△	△	○	A,B,C
E. wtap80211d & TAP デバイス	○	△	○	A,B,C
F. netpoll	△	○	△	B,C

6.　まとめ

本稿では，無線ネットワーク TAP デバイスを用いた無線 LAN エミュレーションフレームワークに，既存のネットワークシミュレータを組み込む際に課題となるエミュレーションフレームワーク内におけるシミュレータ機能の配置方法と，ネットワークアプリケーションとシミュレータ間で無線 LAN フレームと制御情報を交換する方法を議論した．設計したシミュレータ機能の配置方法とアプリケーションとシミュレータ間の情報交換の方法の比較により，実機利用の必要性がなく，エミュレーション規模が大きくない場合には，シミュレータ機能とネットワークアプリケーションを同一ホスト上で実行し，wtap80211d (API) とネットワーク TAP デバイスを併用して情報を交換する方法が適していることを示した．今後，本稿で設計したインタフェースを実装し，エミュレーション実行時のリアルタイム性の検証を行う予定である．

謝辞

本研究の一部は，文部科学省科学研究費補助金 15H02689 及び 17K20027 の支援の下で行われたものである．ここに記して謝意を示す．

参考文献

[1] A. Kato, M. Takai, S. Ishihara: Design and Implementation of a Wireless Network Tap Device for IEEE 802.11 Wireless Network Emulation, In Proc. of ICMU2017, pp.107–112, 2017.

[2] 加藤新良太, 高井峰生, 石原進: 無線ネットワーク TAP デバイスを用いた無線 LAN エミュレータの設計, DICOMO2017, pp.1808–1816, 2017.

[3] ns-3: `https://www.nsnam.org`, (2018 年 5 月 1 日確認).

[4] Space-Time Engineering, Inc: Scenargie, `https://www.spacetime-eng.com/en/products?page=product\_visuallab#base\_simulator`, (2018 年 5 月 1 日確認).

[5] ns-3: Direct Code Execution: `https://www.nsnam.org/overview/projects/direct-code-execution/`, (2018 年 5 月 1 日確認).

[6] H. Tazaki, F. Urbani, E. Mancini, M. Lacage, D. Câmera, T. Turletti, W. Dabbous: Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments, In Proc. of CoNEXT 2013, pp. 217–228, 2013.

[7] B. Lantz, B. Heller, and N. McKeown. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks, In Proc. of HotNets-IX, pp.1–6, 2010.

[8] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg: Mininet-WiFi: Emulating Software-Defined Wireless Networks, In Proc. of CNSM '15, pp.384–389, 2015.

[9] veth - Virtual Ethernet Device, `http://man7.org/linux/man-pages/man4/veth.4.html`, (2018 年 5 月 1 日確認).

[10] A Linux Foundation Collaborative Project: Open vSwitch, `http://www.openvswitch.org`, (2018 年 5 月 1 日確認).

[11] A. Varga and R. Hornig: An overview of the OMNeT++ simulation environment, In Proc. of SIMUtools '08, pp.1–10, 2008.

[12] T. Staub, R. Gantenbein, T. Braun: VirtualMesh: An Emulation Framework for Wireless Mesh Networks in OMNeT++, In Proc. of SIMUtools '09, pp.1–8, 2009.