情報処理学会論文誌 Vol.60 No.1 27–37 (Jan. 2019)

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開発

加藤 新良太[1,a)] 高井 峰生[2,3,b)] 石原 進[1,c)]

受付日 2018年4月10日, 採録日 2018年10月2日

概要：車車間通信等の多数の通信端末で構成される無線 LAN システムの研究・開発において，実際にシステムを構築せずにその動作や性能を評価する方法の 1 つにネットワークエミュレーションがある．一部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 Ethernet デバイスを用いてエミュレーション機能を実装している．ネットワーク TAP デバイスは，OS と Ethernet デバイス間で交換されるパケットデータをユーザ空間に転送できるが，無線 LAN フレームや，送信電力や BSSID 等の無線 LAN デバイスの制御情報を扱えない．このため，ETSI ITS-G5 DCC 等のクロスレイヤ制御をともなう無線 LAN システムの評価が困難である．そこで，本稿では，ユーザアプリケーションとの間で無線 LAN フレームと無線 LAN デバイスの制御情報の双方を交換できる無線ネットワーク TAP デバイスを提案する．また，無線ネットワーク TAP デバイスを用いたエミュレーションフレームワークの設計・実装について述べる．性能評価から既存の無線 LAN システムのソフトウェアを実環境と同等に評価可能であることを示す．

キーワード：無線 LAN エミュレータ，IEEE 802.11，無線ネットワーク TAP デバイス，Linux

Development of a Wireless LAN Emulation Framework based on Wireless Network Tap Device

Arata Kato[1,a)] Mineo Takai[2,3,b)] Susumu Ishihara[1,c)]

Received: April 10, 2018, Accepted: October 2, 2018

Abstract: Network emulation is one of techniques to evaluate the operation and the performance of wireless network systems such as vehicle-to-vehicle communication systems. Some network simulators support a network emulation function leveraging a virtual Ethernet device called *network tap device*. Network tap devices can capture data packets from the protocol stacks of an operating system and send/receive the packets to/from a user application. However, it is difficult to emulate the behavior of wireless network systems with cross-layer management such as ETSI ITS-G5 DCC using existing network tap devices because network tap devices do not capture wireless LAN frames, control parameters of wireless devices such as transmission power, signal strength, etc. Therefore, in this paper, we propose an alternative virtual device we call wireless network tap device (wtap80211) that enables to capture wireless LAN frames and control parameters that existing network tap devices do not capture, and we describe design and implementation of a wireless LAN emulation framework based on the proposed wireless network tap device. We demonstrate that our framework can be used to evaluate the operation of a wireless network system based on experiments with the prototype of the system.

Keywords: wireless LAN emulator, IEEE 802.11, wireless network tap device, Linux

[1] 静岡大学
Shizuoka University, Hamamatsu, Shizuoka 432–8561, Japan
[2] 大阪大学
Osaka University, Suita, Osaka 565–0871, Japan
[3] カリフォルニア大学ロサンゼルス校
University of California, Los Angels UCLA, Los Angels, CA, U.S.A.

[a)] kato.arata.17@shizuoka.ac.jp
[b)] mineo@ieee.org
[c)] ishihara.susumu@shizuoka.ac.jp

1. はじめに

無線 LAN システムの研究・開発において，実環境上に無線 LAN を構築せずにシステムの動作や性能を評価できる環境があれば便利である．そのような評価方法には，数学モデルによる解析 [1]，シミュレーション [2]，エミュレーション [3], [4], [5] がある．数学モデルによる解析では，短時間で評価結果を算出できるが，ハード・ソフトウェアの動作，電波伝搬等の無線 LAN システムの振舞いすべてを数学モデルに表すことが難しい．シミュレーションは，シミュレーションモデルを用いることで元のシステムの動作を模擬するが，評価結果の忠実性はシミュレーションモデルやその実装に依存する．一方，エミュレーションは，実際のシステムの一部を実環境と同様に実行しつつ，電波伝搬等のシステムの動作に影響する事象をシミュレーションモデルで模擬できる．

ns-3 [2]，EXata [3]，Scenargie [5] 等の既存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 [6] と呼ばれる仮想 Ethernet デバイスを用いてネットワークエミュレーション環境を構築する機能を有している．それらの機能では，オペレーティングシステム（OS）のネットワークプロトコルスタックとネットワークシミュレータを接続し，ユーザアプリケーション間で送信・受信されるパケットに対して，転送遅延や帯域制限を模擬する．

図 1 は，ネットワーク TAP デバイスの動作例を表している．ネットワーク TAP デバイスは，OS に対して Ethernet デバイスのように振る舞い，あるユーザアプリケーションからプロトコルスタックを介してパケットを受け取ると，そのデータパケットを別のユーザアプリケーションへ直接転送する．このため，ユーザアプリケーションから見て，実際は Linux のメモリ空間を介して交換されているパケットを Ethernet デバイスを介してパケットが交換されているように見せる．

しかしながら，多くの無線 LAN システムでは，データパケットの送信・受信に加えて，変調や送信電力制御，BSSID の交換等の通信制御を必要とする．しかし，ネットワーク TAP デバイスを利用する既存のエミュレーション環境では，ネットワーク TAP デバイスが有線 LAN のみに対応するため，送信電力や RSSI（Receive Signal Strength Indication）等の無線 LAN の制御情報をネットワークシミュレータと OS の間でやりとりできない．このため，クロスレイヤ制御をともなう無線 LAN システムを実環境と同様に動作させることは難しい．たとえば，欧州電気通信標準化機構（ETSI）で標準化された車車間通信制御プロトコル（ETSI ITS-G5 DCC）[7] では，車両が送信するビーコンの衝突回避のため，各車両は他車両から受信したビーコンの RSSI から自車両周辺の車両密度を推定し，ビーコンの送信頻度や送信出力を動的に変える．

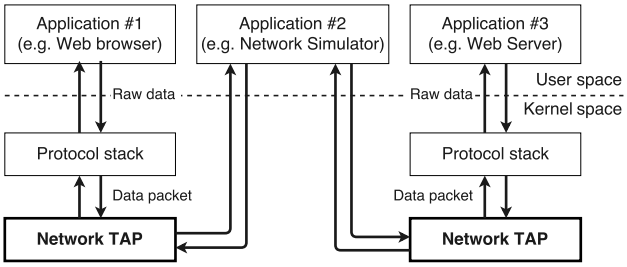


図 1 ネットワーク TAP デバイスの動作例
Fig. 1 Example of the operation of network tap device.

そこで，本稿では，Linux 向けの無線 LAN に対応した無線ネットワーク TAP デバイス（wtap80211）を提案する．無線ネットワーク TAP デバイスは，Linux システムに対して無線 LAN デバイスとして振る舞い，無線 LAN フレームと送信電力や SSID 等の無線 LAN の制御情報の双方を，ネットワーク TAP デバイスと同様にユーザアプリケーションとの間で送信・受信できる仮想無線 LAN デバイスである．本稿では，Linux 向けの無線 LAN エミュレーションフレームワークの設計および実装と，その性能評価について述べる．Linux システムで実際に動作するネットワークアプリケーションを用いた性能評価により，Linux 向けに実装された無線 LAN システムのソフトウェアの動作を，ソースコードの改変やその内部の処理を抽象化することなく評価可能であることを示す．

以下，2 章で既存のネットワークエミュレータと wtap80211 の先行技術について述べ，3 章で本稿で提案する無線ネットワーク TAP デバイスの設計について述べる．4 章で無線ネットワーク TAP デバイスを用いたエミュレーションフレームワークの説明と実装について述べ，5 章でその性能を評価する．6 章で本稿のまとめを述べる．

2. 関連研究および先行技術

ネットワークエミュレーションの実現方法は，評価対象となる無線 LAN システムを抽象化する方法とその対象によって様々なものがある．以下，無線 LAN に着目して，抽象化の対象別に既存のネットワークエミュレーション環境の特徴とその課題について述べる．

2.1 パケットフィルタを用いるエミュレーション環境

Dummynet は，BSD 系 OS に標準で搭載されているパケットの転送遅延や帯域制限を模擬するネットワークエミュレータ [8] で，OS 内部のファイアウォールツール（ipfw）が持つパケットフィルタ機能を利用する．このネットワークエミュレータは，特定のネットワークインタフェースを監視し，そのインタフェースから送信・受信されるデータパケットに対して転送遅延やパケットロス，帯域制限を模擬できる．これにより，パケットの転送遅延やパケットロス，帯域制限ある場合のネットワークアプリケーショ

ンの動作を評価できる．しかしながら，Dummynet は，データリンク層以下の挙動を模擬しないため，Ethernet や IEEE 802.11 等のデータリンク層以下の通信規格の違いを考慮に入れた評価ができない．また，Linux 系 OS においても netem [9] と呼ばれる Dummynet と同様の機能を持つトラフィック制御の仕組みがある．

2.2 仮想化技術を用いたエミュレーション環境

EMANE（Extendable Mobile Ad-hoc Network Emulator）は，IEEE 802.11 におけるモバイルアドホックネットワーク（MANET）上で動作するアプリケーションの挙動を評価できるネットワークエミュレータである [10]．EMANE は，コンテナ型仮想化技術を利用し，多数の通信ノードによる無線 LAN 通信を 1 台あるいは少数のホストマシンで模擬できる．しかしながら，EMANE は，アプリケーションから送信されたパケットをネットワークシミュレータに渡すことで，MAC 層以下におけるデータパケットの挙動を模擬するため，OS による無線 LAN デバイスの操作や制御を模擬できない．

Kawai らは，仮想ハードウェアを利用したネットワークエミュレーション環境 HiFEE（High Fidelity Emulation Environment）を提案している [12]．HiFEE は，実在する無線 LAN デバイス（Atheros 製 AR9160）の動作を忠実に模擬する仮想ハードウェアを搭載した複数の仮想マシンどうしで，ホスト OS 上のネットワークシミュレータを介して無線 LAN フレームを交換することで IEEE 802.11 における無線通信を模擬する．この仮想ハードウェアは，OS から送信される命令を実機と同様に解釈・実行するため，Linux 向けの無線 LAN デバイスドライバ（ath9k）[13] がエミュレーション環境上で実環境と同様に動作する．しかし，仮想マシンの実行には，ホストマシンに高性能な計算処理能力が要求されるため，多数の通信ノードを同時に模擬すると実時間性が低下する．

明石らは，データリンク層以下を抽象化するネットワークエミュレータ Meteor を提案している [14]．Meteor は，障害物やネットワークノードの位置等のユーザが任意に定義する無線通信環境のシナリオから，各ノード間の信号品質を表すパラメータ（フレームロス率，遅延時間等）を算出し，そのパラメータに基づいて Ethernet フレームに対してフレームロスや転送遅延を発生させる．

また，Akashi らは，Meteor を利用した無線ネットワークエミュレーション環境 NETorium を提案している [15]．NETorium は，Ethernet フレームにカプセル化された無線 LAN フレームに対して転送遅延を発生させる．このため，エミュレーション環境の構築に Ethernet で接続された有線 LAN を構築する必要があり，明石らの実装では，StarBED [16] と呼ばれる 1,000 台以上の計算機サーバで構成されるテストベッド設備を利用している．

真野らは，IEEE 802.11 や LTE，TV ホワイトスペースで利用される周波数帯における電波伝搬を正確に模擬できる無線システムエミュレータを提案している [17]．無線システムエミュレータは，主にネットワークノードの模擬に用いる仮想マシンと電波伝搬を模擬する空間エミュレータで構成される．このシステムは，実際の無線デバイスが信号の送信過程で生成するデジタル I/Q データを基に電波伝搬をシミュレータで模擬する．無線システムエミュレータは，実時間性を確保するため，電波伝搬の模擬に十分に高速な計算処理装置を必要とする．真野らの実装では，FPGA を用いて電波伝搬の計算を行っている．

2.3 仮想デバイスを用いるエミュレーション環境

既存研究の一部には，ネットワーク TAP デバイスに類似した，無線 LAN フレームをユーザ空間に転送する機能を持つ仮想デバイスを用いてエミュレーション環境を構築するものがある．Weingärtner らは，Linux の net_device を利用した無線 LAN エミュレーション環境を提案している [18]．net_device は，Linux における有線 LAN デバイスおよび無線 LAN デバイスを含むネットワークデバイスを抽象化する API である．このエミュレーション環境では，net_device を利用して ns-3 [2] と Linux システムを接続し，シミュレータが模擬した無線 LAN システムの MAC 層および物理層の挙動を反映したシミュレーション結果を net_device を通じて受信信号強度やデータレートとして参照できる．しかし，その実装には，現在の Linux で用いられている Linux Wireless Subsystem [19] 以前の実装である Linux Wireless Extensions [20] が用いられているため，IEEE 802.11n/ac/ad や IEEE 802.11p/s 等の新しい無線 LAN 規格を用いる無線 LAN システムを模擬できない．

本稿で議論する無線ネットワーク TAP デバイスに類似する技術に，mac80211_hwsim と呼ばれる Linux 向けの仮想無線 LAN デバイスがある [21]．mac80211_hwsim は，Linux Wireless Subsystem に対して無線 LAN デバイスとして振る舞い，無線 LAN フレームをユーザアプリケーションとの間で送信・受信できる．しかし，mac80211_hwsim は，iw [22] や wpa_supplicant [23] 等の Linux の無線 LAN ツールのデバッグに用いられるため，無線 LAN デバイスとそのデバイスドライバの間で交換される制御情報の一部をユーザアプリケーションとの間でやりとりできない．

mac80211_hwsim がユーザ空間と交換できない無線 LAN デバイスの制御情報には，無線 LAN デバイスの電源や省電力機能の状態を保持するパラメータや，Regulatory domain に基づく国や地域別に定められている送信電力の上限値や利用可能な周波数帯の定義等がある．このため，無線 LAN デバイスの状態やデバイスの操作・制御を必要とするネットワークアプリケーションや無線 LAN システムの動作を mac80211_hwsim で模擬することは難しい．

2.4 既存のエミュレーション環境の課題

既存のエミュレーション環境は，大きく分けて通信ノード1台あたりの挙動を忠実に模擬するエミュレータ（HiFEE，無線システムエミュレータ）と，多数のノードで構成される大規模なネットワークの挙動を模擬するエミュレータ（EMANE，NETorium）に分類できる．前者は，通信ノード単体の挙動の再現性が高いが，その挙動を模擬するために高性能な計算機を必要とするため規模性が低い．後者は，大規模なネットワークシステムの評価に適しているが，ネットワークノード1台あたりの再現性は前者と比較すると低い．また，両者ともにFPGAや計算機サーバ等を必要とするため，エミュレーション環境の構築にコストや時間がかかる．

Weingärtnerらの仮想デバイスやmac80211_hwsimの既存の仮想無線LANデバイスでは，IEEE 802.11acやIEEE 802.11p等の新しい無線LAN規格への対応が不十分であるため，ETSI ITS-G5 DCC等のクロスレイヤ制御をともなう無線LANシステムを模擬することが難しい．

このため，無線LANシステムの動作を正確に模擬するためには，電力制御や無線LANフレームの送信制御等のシステムによる無線LANデバイスの制御を模擬できることが求められる．また，エミュレーション環境構築におけるコストや時間を削減するため，仮想マシンや大規模なテストベッド設備等を必要としないエミュレーション環境が望まれる．

3. 無線ネットワークTAPデバイス

本章では，本稿で提案する無線ネットワークTAPデバイスのアーキテクチャとその基本動作について述べる．

3.1 無線ネットワークTAPデバイスの概要

無線ネットワークTAPデバイス（wtap80211）は，Linux Wireless Subsystemを利用したLinuxシステムで動作する仮想無線LANデバイスドライバとして実装される．**表1**は，wtap80211が模擬する仮想無線LANデバイスの構成を示している．wtap80211は，Linux Wireless Subsystemを利用する仮想デバイスという点ではmac80211_hwsimと基本構造は同じである．しかし，mac80211_hwsimは，Linux Wireless Subsystemから送信された制御情報をユーザ空間に送信する機能を有していない一方，wtap80211はその機能を有している．

このため，mac80211_hwsimを用いたエミュレーションでは，ネットワークシミュレータ等のユーザプロセスからLinux Wireless Subsystemによる通信制御の様子を把握できないため，シミュレータは，OSやアプリケーションによる送信電力制御や，BSSの更新，受信信号強度の取得等の通信制御の動作を模擬できない．一方，wtap80211を用いたエミュレーションでは，それらの通信制御の様子をシミュレータが把握できるため，OSやアプリケーションによる通信制御の動作も模擬できる．また，wtap80211は，Linuxシステムで実際に動作するath10kやwil6210等の無線LANデバイスドライバの実装を参考に，IEEE 802.11ac/adやIEEE 802.11pでの動作時に必要な利用周波数帯のリストや，マルチチャネルでの動作時のチャネルコンテキストの切替え動作の定義を追加することで，最新の無線LAN規格に対応しつつ，無線LANフレームに加えて，送信電力やBSSID，受信信号強度等の無線LANの制御情報をネットワークシミュレータ等のユーザプロセスとの間で交換できるよう設計・実装されている．また，wtap80211は，ユーザ空間との間にLinux Wireless Subsystemを介さないデータパスを持つ．このため，無線LANフレームの送信・受信に加え，Linux Wireless Subsystemが発信した無線LANデバイスの制御命令とそれ付随する制御パラメータをユーザ空間に直接通知できる．

表1 wtap80211の仮想無線LANデバイスの構成
Table 1 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

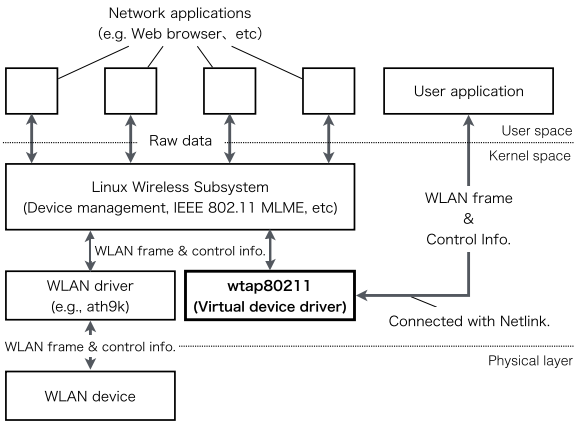


図2 wtap80211のアーキテクチャ
Fig. 2 Architecture of wtap80211.

3.2 基本動作

wtap80211は，Linuxにおいて仮想無線LANドライバとして実装されており，既存無線LANドライバと同様のAPIを用いて，Linux Wireless Subsystemと無線LANフレームや無線LANデバイスの制御パラメータをやりとりする．**図2**はwtap80211のアーキテクチャを示している．Linux Wireless Subsystemは，Linuxシステムにおいて無線LAN通信時におけるアソシエーションや認証等のノード間の接続処理や，無線LANデバイスの管理等を担うシステムである．wtap80211は，Linux Wireless Subsystemから無線

LAN フレームや無線 LAN デバイスの制御命令を受け取ると，ユーザ空間のプロセスに向けて Netlink メッセージを送信する．Netlink は，Linux システムにおいて，プロセス間通信もしくはユーザプロセスとカーネルモジュール間の通信に用いられる API である [26], [27].

wtap80211 は，Linux Wireless Subsystem から無線 LAN フレームや制御情報を受け取った場合，Netlink を経由してユーザ空間にそれらのデータを送信する．また，ユーザ空間から Netlink を経由して無線 LAN フレームや受信信号強度等を受け取った場合は，それらのデータを Linux Wireless Subsystem に渡す．この際に，Linux Wireless Subsystem は，実際に無線 LAN 通信をしている場合と同様に無線 LAN フレームの送信・受信処理やアソシエーション，認証等の処理を実行する．したがって，wtap80211 は，端末の移動にともなう受信信号強度の変化に応じて生じたアクセスポイントへの接続・離脱，アドホックネットワークにおける動的なトポロジの変化等を再現可能である．

Linux では，Netlink 以外にもユーザプロセスとカーネルモジュール間でデータを交換する仕組みとして，システムコール，ファイルシステム，ソケット API を利用する方法がある．システムコールは，呼び出し時の処理が煩雑になりやすく，カーネルパニック等の深刻な不具合を誘発しやすい．ファイルシステムは，ファイルの読み書きにともなうオーバヘッドが大きく，データの読み書きを頻繁行うフレームの送信・受信処理には向いていない．また，ソケット API は，ネットワークプロトコルスタック以外のカーネルモジュールからパケットデータを参照できない．一方，Netlink は，ソケット API を用いてカーネルモジュールと通信でき，プログラムの記述が容易なため，カーネルパニック等の深刻な不具合を誘発しにくい．また，メモリ上にソケット API とは独立したメッセージキューを持ち，データの読み書きにともなうオーバヘッドがファイルシステムより小さい．

wtap80211 とユーザプロセス間でやりとりされるメッセージには，フレームデータとそのフレームの送信に必要な制御情報が含まれるフレームメッセージと，制御命令とその命令の実行に必要な制御パラメータが含まれる制御メッセージの 2 種類がある．**図 3** と**図 4** は，フレームメッセージと制御メッセージのフォーマットを表している．フレームメッセージと制御メッセージ双方の先頭 20 Bytes（nlhdr と genlhdr）は，Netlink ソケットのプロトコルヘッダである．Netlink ソケットのプロトコルヘッダの直後にある 4 Bytes 長の領域は，wtap80211 の内部で使用するために予約された領域である．

フレームメッセージには，フレームデータに加え，フレームの送信・受信に必要となる制御パラメータが格納される．wtap80211 からユーザ空間へ無線 LAN フレームを転送する場合，wtap80211 は，Linux Wireless Subsystem から sk_buff 構造体を受け取り，sk_buff 構造体に格納されている無線 LAN フレームをフレームメッセージの raw data に格納する．また，sk_buff 構造体のコントロールバッファ（cb）に格納されている送信電力や送信周波数，帯域幅等から構成される送信制御情報をフレームメッセージの Control info に格納した後，そのメッセージをユーザ空間に転送する．

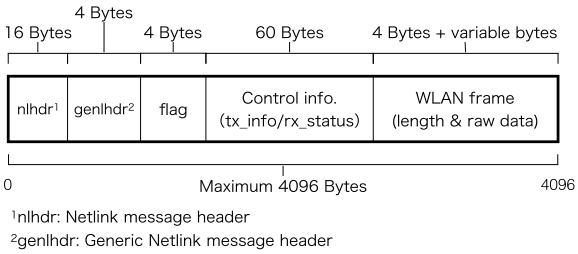


図 3 フレームメッセージのフォーマット
Fig. 3 Format of the frame message.

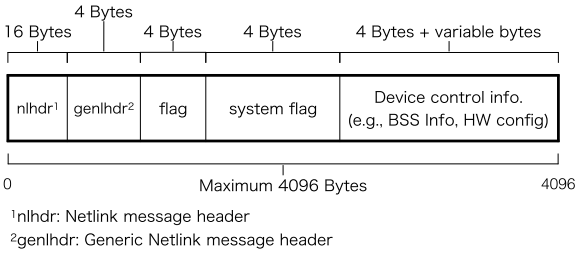


図 4 制御メッセージのフォーマット
Fig. 4 Format of the control message.

ユーザ空間から無線 LAN フレームを受け取る場合には，wtap80211 は，新たに確保した sk_buff 構造体に，フレームメッセージの raw data に格納されている無線 LAN フレームを格納する．また，フレームメッセージの Control info に格納されている受信信号強度や受信周波数等で構成される受信時の状態情報を cb に格納した後，Linux Wireless Subsystem にその sk_buff 構造体を渡す．また，IEEE 802.11n/ac で送信・受信される場合は，MCS（Modulation and Coding Scheme）のインデックス番号等が送信制御情報や受信状態情報に追加される．

制御メッセージには，送信された制御命令の種別に合わせて異なる情報が格納される．Linux Wireless Subsystem から送信される制御情報は，a) Basic Service Set（BSS）情報，b) 無線 LAN デバイスが持つ送信キューの設定情報，c) 受信フレームのフィルタリングルール，d) 無線 LAN デバイスのハードウェアの設定情報等の複数のグループに大別され，グループごとに関連するパラメータが集約されている．グループ内のパラメータが以前から 1 つでも変更されると，Linux Wireless Subsystem は，グループ単位で制御情報とグループ内の変更されたパラメータを識別するフラグを wtap80211 に送信する．図 4 の制御メッセージに含まれる system flag は，そのパラメータを識別するフラ

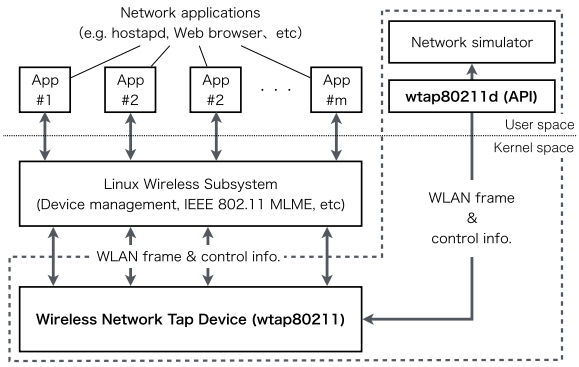


図 5 無線 LAN エミュレータの概要
Fig. 5 Outline of a WLAN emulator based on wtap80211.

グをコピーして格納したものである．

4. 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，3 章にて述べた無線ネットワーク TAP デバイスを利用した無線 LAN エミュレーションフレームワークの概要およびその設計について述べる．

4.1 要求要件

2 章で述べた既存のネットワークエミュレーション環境は，ユーザアプリケーションを実環境と同様に実行できる．しかしながら，それらの多くは，データ通信を模擬することを目的としており，アプリケーションによる無線 LAN デバイスの制御を模擬できない．また，エミュレーション環境の構築に高性能もしくは大規模な計算機設備を必要とするため，環境構築のためにコストや時間がかかることが課題となる．このため，i) 多数の通信ノードで構成される無線 LAN システムの動作を模擬すること，ii) エミュレーション環境の構築にあたり，仮想マシンや多数の計算機で構成される大規模なテストベッド設備等を必要としないこと，iii) OS やユーザアプリケーションによるデバイス制御を模擬できることの計 3 つの要件を満たすエミュレーション環境を構築した．

4.2 基本設計

本稿で設計・実装した無線 LAN エミュレーションフレームワークは，wtap80211 がインストールされた Linux システムと，その Linux システム上で動作する API（wtap80211d）で構成される．**図 5** に無線 LAN エミュレーションフレームワークの概要を示す．このエミュレーションフレームワークは，MAC 層における IEEE 802.11 MLME（MAC-Sublayer Management Entity）とそれより上層の処理を Linux カーネルを用いて行い，キャリアセンスと物理層における振舞いは ns-3 や Scenargie 等の既存のネットワークシミュレータで模擬する．

表 2 wtap80211d の内部 API の一覧
Table 2 List of internal APIs of wtap80211d.

関数名	機能
genl_init()	Netlink コネクションを作成する
genlmsg_unicast_custom()	wtap80211 へフレームメッセージや制御メッセージ等の Netlink メッセージを送信する
genlmsg_recv_mgmt_msg()	wtap80211 から制御メッセージを受信する
genlmsg_recv_frame_msg()	wtap80211 からフレームメッセージを受信する

ネットワークシミュレータを用いた抽象化部分では，wtap80211 を利用して複数台の仮想無線 LAN デバイスを構築し，1 つの Linux システム上に複数の無線 LAN 端末が存在する環境を模擬する．各仮想無線 LAN デバイスは，Netlink ソケットで接続された API を介して wtap80211d と接続し，ネットワークシミュレータとの間で無線 LAN フレームや無線 LAN デバイスの制御情報をやりとりする．

Linux システムが動作する非抽象化部分では，Linux Wireless Subsystem とプロトコルスタック，ユーザアプリケーションが動作する．このため，Linux Wireless Subsystem を介して無線 LAN デバイスを制御するアプリケーションであれば，エミュレーション環境上で実環境と同等に実行できる．

wtap80211 が送信した無線 LAN フレームや無線 LAN デバイスの制御パラメータは，wtap80211d（API）を通してユーザアプリケーションに渡される．wtap80211d は，C 言語で記述されており，ネットワークシミュレータを含むユーザアプリケーションに対する API として動作する．**表 2** は，wtap80211d に実装した API の一覧である．wtap80211 からメッセージを受信した場合は，そのメッセージに含まれるフレームデータや無線 LAN デバイスの制御パラメータを解析し，ユーザアプリケーションにそれらのデータをわたす．また，ネットワークシミュレータから wtap80211 に対してメッセージの送信要求が出された場合，送信するデータの内容に合わせたメッセージを作成し，wtap80211 に送信する．

図 6 は，エミュレーションフレームワーク上で，hostapd（Linux マシンをアクセスポイントとして動作させるアプリケーション）を起動してビーコンを送信した後，hostapd を終了した際に，無線ネットワーク TAP デバイスと wtap80211d 間でやりとりされた無線 LAN フレームや制御情報を，wtap80211d のログファイルの出力結果から確認できていることを表す図である．図 6 では，hostapd からビーコンフレームが送信された様子に加えて，hostapd が終了する際に送信した Deauth フレームと，その際に無線 LAN デバイスのビーコン送信機能の無効化と，デバイスステータスをアイドル状態に変更した様子が確認できる．

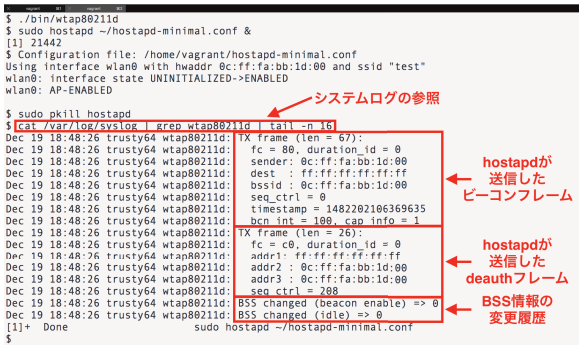


図 6 wtap80211d のログファイルの出力結果
Fig. 6 Log output of wtap80211d.

4.3 設計したフレームワークのオーバヘッド

本エミュレーションフレームワークでは，無線 LAN システムとエミュレーション環境が同一の Linux システムで動作し，かつ無線 LAN システムは実時間で動作する．このため，無線 LAN フレームを送信・受信する際にかかるオーバヘッドが無線 LAN システムの性能に直接影響する．

本エミュレーションフレームワークにおけるオーバヘッドの主な要因は，ネットワークシミュレータと wtap80211 間の Netlink メッセージ転送にかかる遅延と，ネットワークシミュレータの処理遅延の 2 つが考えられる．

wtap80211 は，Netlink メッセージを介してユーザアプリケーションと無線 LAN フレームや無線 LAN デバイスの制御パラメータをやりとりするため，そのメッセージのやりとりには，ユーザ空間におけるメモリ操作が頻繁に行われる．ユーザ空間におけるメモリ操作は，仮想アドレスを介してメモリを扱うため，カーネル空間におけるメモリ操作よりもオーバヘッドが大きい．

ネットワークシミュレータの処理遅延は，ネットワークシミュレータの実装やシミュレーションに用いるモデルの計算量に依存する．このため，シミュレーション処理のオーバヘッドは，利用するネットワークシミュレータやシステムモデルごとにそのオーバヘッドを計測することが必要と考えられる．ユーザアプリケーション間の転送遅延を実環境と同等に再現することを考えた場合，ネットワークシミュレータの処理遅延と Netlink メッセージの転送遅延を足した遅延時間が実環境におけるフレームの転送遅延よりも短くなることが好ましい．このため，本エミュレーションフレームワークにおける wtap80211 と wtap80211d（API）間における Netlink メッセージの転送遅延が，実環境におけるフレームデータの転送遅延よりも短くなることが求められる．

5. オーバヘッドの評価

本章では，本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，シミュレータ部分を除いたフレームワークのオーバヘッドが十分に小さく，シミュレータによる伝送遅延の挿入やパケットロス等の模擬により，無線 LAN の挙動のエミュレーションを行う余地があるかを明らかにするため，4 章にて述べた wtap80211 と wtap80211d（API）間における Netlink メッセージの転送遅延を評価した．評価に用いた計算機の概要を**表 3** に示す．計算機の性能差によるオーバヘッドの大小を比較するため，性能が異なる 2 台の計算機（Machine 1，Machine 2）を用意した．また，Linux カーネルのチューニングの違いによる性能差を考慮し，各計算機でデスクトップ PC 等で一般的に用いられるカーネル（generic カーネル）と，リアルタイムカーネル（lowlatency カーネル）をそれぞれ用いた場合の性能も評価した．generic カーネルと lowlatency カーネルの違いは，タイマー割り込みの周期を generic カーネルでは 250 Hz（4 ms）に対し，lowlatency カーネルでは 1,000Hz（1ms）とし，また，lowlatency カーネルに対してプロセスコンテキストにおいてカーネルプロセスが実行中でも他のプロセスに処理を切り替えるプリエンプション機能を有効にした．

表 3 評価環境の計算機性能
Table 3 Machine specification.

	Machine 1	Machine 2
OS	Linux 3.19.8	Linux 3.19.8
Distribution	Ubuntu 14.04.5 LTS	Ubuntu 14.04.5 LTS
CPU	Intel Celeron N2830	Intel Core i5 3230M
メモリ	DDR3-1366 2 GB	DDR3-1600 8 GB
ストレージ	SSD 16 GB	HDD 350 GB
チップセット	Intel Centrino Advanced-N 6205	N/A
ビットレート	300 Mbps	N/A
対応規格	IEEE 802.11a/b/g/n	N/A

評価指標は，通信ノード間のスループットと往復遅延時間（RTT）とした．この評価では，ネットワークシミュレータを除いたエミュレーションフレームワーク上の模擬端末間で，CSMA/CA によるメディアアクセス制御や電波伝搬遅延等を考慮せずにデータ通信を行った場合の最大スループットと往復遅延時間を測定した．スループットは iperf3 を用いて測定し，RTT は ping コマンドを用いて測定した．**表 4** に，測定時におけるエミュレーション環境上のネットワークインタフェースのパラメータと，スループット測定時のパラメータを示す．なお，エミュレーション環境における各評価指標の測定では，24 台の仮想無線端末をサーバとクライントからなる 2 台 1 組の計 12 組に分け，それぞれの組ごとに UDP ストリームを 1 つ作成し，異なるチャネルで並行して通信するものとした．

表 4 測定ツールに与えたパラメータ
Table 4 Parameters of measurement tools.

Network interface	動作モード	Ad-hoc
	無線 LAN 規格	IEEE 802.11a
	チャネル	36（5,180 MHz）
	ビットレート	54 Mbps
iPerf3	L4 プロトコル	UDP
	バッファサイズ	1470 bytes
	ビットレート	54 Mbps
	Omit time	10 sec
ping	送信間隔	1 packet/sec
	パケットサイズ	64 bytes

表 5 スループットの測定結果
Table 5 Emulated throughputs.

UDP ストリーム数	平均スループット [Mbps]			
	Machine 1		Machine 2	
	generic	lowlatency	generic	lowlatency
2	53.49	53.92	53.91	53.92
4	53.52	53.92	53.91	53.91
6	53.51	53.99	53.91	53.91
8	53.56	53.93	53.92	53.92
10	53.51	46.75	53.90	53.91
12	53.06	49.22	53.91	53.92

5.1 スループット

表 5 に，各計算機で generic カーネルおよび lowlatency カーネルを用いて測定したスループットの測定結果を示す．Machine 1 では，generic カーネルを用いて計測した場合，同時に 12 台の通信ノードが通信する場合にスループットが少し減少したものの，53.50 Mbps のスループットを得られた．lowlatency カーネルを用いて計測した場合は，通信ノードの数が 8 台までは generic カーネルで計測したスループットよりも高い約 53.90 Mbps のスループットが得られたが，通信ノードの数が 10 台を超えると，スループットが 50 Mbps を下回る結果となった．一方，Machine 2 を用いたスループットの測定では，generic カーネルおよび lowlatency カーネルのいずれを用いた場合においても通信ノードの数の増減にかかわらず，約 53.90 Mbps のスループットを得られた．ここで，現実の場合での IEEE 802.11a の動作では，複数の端末が同一チャネルで通信している場合のスループットは，理論値よりも小さく，50Mbps 以上を記録することはないが，本実験では，CSMA/CA によるメディアアクセス制御や電波伝搬遅延，伝送時間等のシミュレーション部分を除外しているため，それらに起因する通信速度の制限はないことに注意されたい．

Machine 1 で端末台数の増加にともないスループットが低下した理由は，通信ノード数が増加したことで交換されるパケット数が増えたためと考えられる．また，lowlatency カーネルにおいてスループットが大きく減少した理由は，

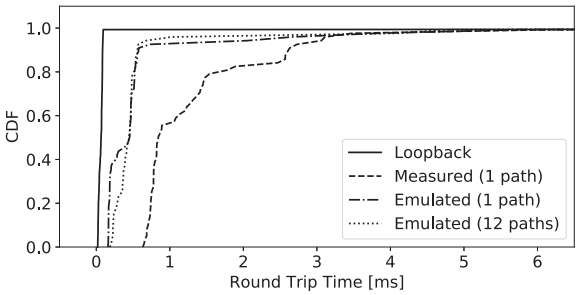


図 7 generic カーネルを用いた場合の Machine 1 の RTT
Fig. 7 RTTs of Machine 1 with the generic kernel.

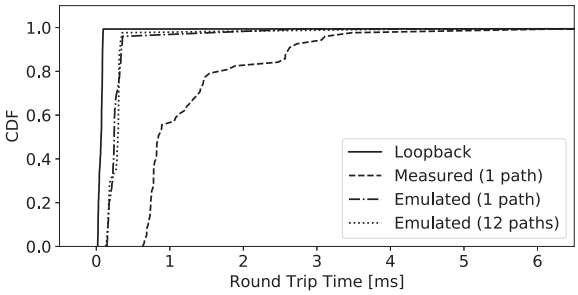


図 8 generic カーネルを用いた場合の Machine 2 の RTT
Fig. 8 RTTs of Machine 2 with the generic kernel.

プリエンプション機能により，パケットの送信・受信処理を担うカーネルプロセスの実時間が他のプロセスの処理待ちのため generic カーネルと比較して長くなったため，パケット処理にともなうオーバヘッドが増大したと考えられる．

以上より，計算機能力に余裕があれば，処理能力が低い計算機においても複数の通信ノードによる無線 LAN 通信を模擬できるものと考えられる．ただし，IEEE 802.11 において，隠れ端末の関係にある場合を除けば，CSMA/CA により同一チャネル上で複数の端末がある時刻に同時に通信することは基本的にないことを考慮すると，Machine 1 のような計算機性能が低い場合においても，エミュレーション環境上で無線 LAN におけるデータ通信を模擬するために十分なスループットを得ることができると考えられる．

5.2 往復遅延時間

図 7 と**図 8** に generic カーネルをインストールした Machine 1 および Machine 2 において測定した RTT の累積分布を示し，**図 9** と**図 10** に lowlatency カーネルをインストールした Machine 1 および Machine 2 において測定した RTT の累積分布を示す．Loopback デバイスで測定した RTT は，計測に用いた計算機の性能限界を表しており，RTT の累積分布が Loopback デバイスで測定した RTT の累積分布に近づくほどオーバヘッドが小さいことを表す．RTT の実測値は，周囲に 2.4 GHz 帯と 5.0 GHz 帯の無線 LAN アクセスポイントが 20 台混在する室内において，アンテナ間距離が 50 cm となるように 2 台の Machine 1 を

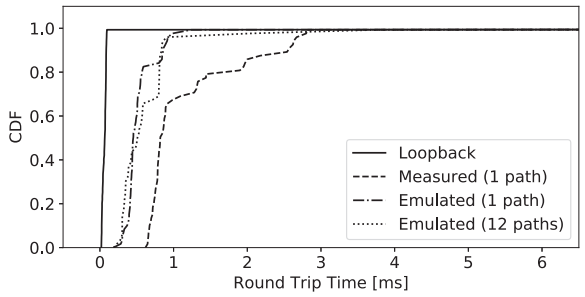


図 9 lowlatency カーネルを用いた場合の Machine 1 の RTT
Fig. 9 RTTS of Machine 1 with the lowlatency kernel.

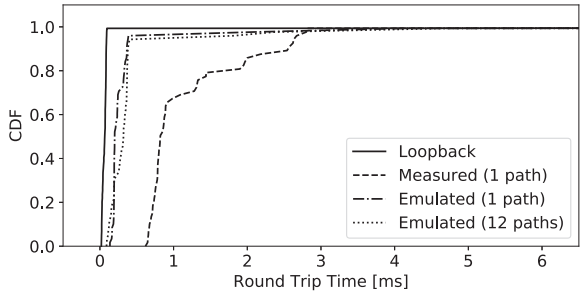


図 10 lowlatency カーネルを用いた場合の Machine 2 の RTT
Fig. 10 RTTS of Machine 2 with the lowlatency kernel.

配置し，その 2 台で IEEE 802.11a アドホックネットワークを構築して測定した．エミュレーション環境で測定した RTT は，2 台のノード間のみで通信した場合と，24 台すべてのノード間で同時に通信した場合の 2 つにおいて測定した．

図 7 および図 8 より，計算機性能が高い Machine 2 の方が Machine 1 よりも RTT が小さい傾向にあることが分かる．また，Machine 1 および Machine 2 のどちらにおいても，通信するノードの数による RTT のばらつきや長さに大きな差は見られなかった．

図 9 および図 10 より，generic カーネルをインストールした計算機での RTT の傾向と同様に，lowlatency カーネルをインストールした計算機においても，計算機性能が良い Machine 2 の方が Machine 1 よりも RTT が小さい傾向にあり，Machine 1 および Machine 2 においてもノード数による RTT のばらつきや長さに大きな差は見られない．

図 7 と図 9 を比較すると，Machine 1 においては，lowlatency カーネルの RTT が，generic カーネルの RTT よりも長くなる傾向にある．一方で，図 8 と図 10 を比較すると，計算機性能が良い Machine 2 では，カーネルのチューニングの違いによる RTT のばらつきや長さに差はほとんど認められないことが分かる．

以上より，シミュレーションによる処理遅延を考慮すると，エミュレーション環境の構築には，計算機性能の違いにかかわらず generic カーネルのほうがシミュレーション処理に当てられる時間をより長く確保できると考えられる．

6. まとめ

本稿では，無線ネットワーク TAP デバイス（wtap80211）と，無線 LAN エミュレーションフレームワークを提案した．wtap80211 は，既存のネットワーク TAP デバイスでは扱えない無線 LAN フレームと無線 LAN デバイスの制御パラメータ双方をユーザ空間と交換できる．また，本稿で提案したエミュレーションフレームワークは，エミュレーション環境の構築に FPGA や計算機サーバ等の特別なハードウェアを必要とせず，既存の Linux システム上に構築可能である．このため，このエミュレーションフレームワーク上では，Linux システムやネットワークアプリケーションを実環境と同様に動作する．

本フレームワークのシミュレーション部分以外の性能評価の結果，計算機の能力が低い場合においてもエミュレーション処理によるオーバヘッドが実環境と比較して十分に小さいことが確かめられた．これにより，シミュレータの処理遅延に依存しない範囲では，4 章にて定義した本フレームワークが満たすべき 3 要件のうち，i)「エミュレーション環境構築にあたり，仮想マシンや多数の計算機で構成される大規模なテストベッドを必要としないこと」，ii)「多数の通信ノードで構成される無線 LAN システムの動作を模擬可能であること」を確認できた．iii)「OS やユーザアプリケーションによるデバイス制御を模擬できること」に関しては，ネットワークシミュレータと接続した上で今後検証する必要がある．このとき，ネットワークシミュレータ側には，wtap80211d との通信に用いるインタフェースの実装と，wtap80211 が模擬する仮想デバイスの MAC アドレスや送信電力，チャネル等の状態とシミュレータのノード状態をリアルタイムに同期する機能の追加が必要である．

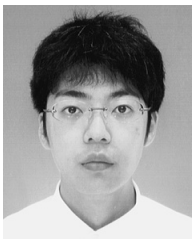
今後，本稿で開発したエミュレーションフレームワークに ns-3 [2]，OMNeT++ [29]，Scenargie [5] 等の既存のネットワークシミュレータを連携させたエミュレータの実装を通じて，CSMA/CA によるメディアアクセス制御や電波伝搬等を考慮した OS やアプリケーションによるデバイス制御を模擬可能であることを確認し，ネットワークシミュレータの処理遅延を含めたエミュレーション処理全体のオーバヘッドを測定する予定である．また，Linux に組み込まれている IEEE 802.11p のプロトコルスタックや，Linux 向けの ETSI ITS-G5 DCC の実装例である OpenC2X [30] を本稿で開発したエミュレーション上で動作させ，車両の移動や電波伝搬の影響を考慮した評価を行う予定である．

謝辞 本研究の一部は，科学研究費補助金 15H02689 および 17K20027 の助成によるものである．

参考文献

[1] Chaing, M., Low, S.H., Calderbank, A.R. and Doyle, J.C.: Layering as Optimization Decomposition: A Math-

ematical Theory of Network Architecture, *Proc. IEEE*, Vol.95, No.1, pp.255–312 (2007).
[2] ns-3, available from ⟨https://www.nsnam.org⟩ (accessed 2018-04-01).
[3] SCALABLE Network Technologies, Inc.: EXata, available from ⟨http://web.scalable-networks.com/exata-network-emulator-software⟩ (accessed 2018-04-01).
[4] Direct Code Execution, available from ⟨https://www.nsnam.org/overview/projects/direct-code-execution⟩ (accessed 2018-04-01).
[5] Space-Time Engineering, LLC.: Scenargie, available from ⟨https://www.spacetime-eng.com/en/products⟩ (accessed 2018-04-01).
[6] Krasnyansky, M.: Universal TUN/TAP device driver, available from ⟨https://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt⟩ (accessed 2018-04-01).
[7] ETSI TS 102 687 v1.1.1 Intelligent Trasnport Systems (ITS), Decentralized Congestion Control Mechanisms for Intelligent Transport Systems Operating in the 5GHz range; Access layer part, ETSI (2011).
[8] Carbone, M. and Rizzo, L.: Dummynet revisited, ACM SIGCOMM Computer Communication Review, Vol.40, No.2, pp.12–20, ACM (2010).
[9] The Linux Foundation: netem, available from ⟨https://wiki.linuxfoundation.org/networking/netem⟩ (accessed 2018-04-01).
[10] U.S. Naval Research Laboratory: The Extendable Mobile Ad-hoc Network Emulator (EMANE), available from ⟨https://www.nrl.navy.mil/itd/ncs/products/emane⟩ (accessed 2018-04-01).
[11] LXC - Linux Containers, available from ⟨https://linuxcontainers.org⟩ (accessed accessed 2018-04-01).
[12] Kawai, T., Kaneda, S., Takai, M. and Mineno, H.: A Virtual WLAN Device Model for High Fidelity Software Emulation, *ACM Trans. Modeling and Computer Simulation*, Vol.27, No.3 (2017).
[13] Linux Wireless: ath9k, available from ⟨https://wireless.wiki.kernel.org/en/users/drivers/ath9k⟩ (accessed 2018-04-01).
[14] 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，電子情報通信学会論文誌，Vol.J98-B, No.4, pp.357–372 (2015).
[15] Akashi, K., Inoue, T., Yasuda, S., Takano, Y. and Shinoda, Y.: NETorium: High-fidelity Scalable Wireless Network Emulator, *12th Asian Internet Engineering Conference* (*AINTEC '16*), pp.25–32, ACM (2016).
[16] 国立研究開発法人情報通信研究機構：StarBED, 入手先 ⟨http://starbed.nict.go.jp⟩（参照 2018-04-01）.
[17] 真野　浩，猿渡俊介：無線システムエミュレータの実装と評価，情報処理学会論文誌，Vol.55, No.5, pp.1541–1554 (2014).
[18] Weingärtner, E., Lehn, H.V. and Wehrle, K.: Device Driver-enabled Wireless Network Emulation, *Proc. 4th International ICST Conference on Simulation Tools and Techniques*, pp.188–197 (2011).
[19] Linux Wireless, available from ⟨https://wireless.wiki.kernel.org⟩ (accessed 2018-04-01).
[20] Tourrilhes, J.: Wireless Extensions for Linux, available from ⟨https://hewlettpackard.github.io/wireless-tools/Linux.Wireless.Extensions.html⟩ (accessed 2018-04-01).
[21] mac80211_hwsim, available from ⟨https://wireless.wiki.kernel.org/en/users/drivers/mac80211_hwsim⟩ (accessed 2018-04-01).
[22] About iw, available from ⟨https://wireless.wiki.kernel.org/en/users/documentation/iw⟩ (accessed 2018-04-01).
[23] Malinen, J.: Linux WPA/WPA2/IEEE 802.1X Supplicant, available from ⟨https://w1.fi/wpa_supplicant/⟩ (accessed 2018-04-01).
[24] IEEE Std 802.11a-1999 (R2003), Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High-speed Physical Layer in the 5 GHz Band, IEEE (2003).
[25] IEEE Std 802.11p(TM)-2010, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Wireless Access in Vehicular Environments, IEEE (2010).
[26] Salim, J., Khosravi, H., Kleen, A. and Kuznetsov, A.: RFC 3549 – Linux Netlink as an IP Services Protocol, IETF, available from ⟨https://tools.ietf.org/html/rfc3549⟩ (accessed 2018-04-01).
[27] Ayuso, P.N., Gasca, R.M. and Lefevre, L.: Communicating between the kernel and user-space in Linux using Netlink sockets, Software – Practice and Experience, Vol.40, No.9, pp.797–810 (2010).
[28] iPerf3: iPerf - The ultimate speed test tool for TCP, UDP and SCTP, available from ⟨https://iperf.fr⟩ (accessed 2018-04-01).
[29] OpenSim, Ltd.: OMNeT++, available from ⟨https://omnetpp.org/omnetpp⟩ (accessed 2018-04-01).
[30] Laux, S., Pannu, G.S., Schneider, S. and Tiemann, J.: Demo: OpenC2X – An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5, *2016 IEEE Vehicular Networking Conference* (*VNC '16*), pp.152–153, IEEE (2016).



加藤 新良太 （学生会員）

1994 年生．2017 年静岡大学工学部数理システム工学科卒業．2018 年同大学大学院修士課程在学中．無線ネットワークエミュレーションに関する研究に従事．

高井 峰生 （正会員）

1997 年早稲田大学大学院博士後期課程修了．2007 年米国法人スペースタイムエンジニアリング設立．現在，カリフォルニア大学ロサンゼルス校主幹開発研究員ならびに大阪大学大学院情報科学研究科招へい准教授．モバイル通信システムおよびその評価方法についての研究に従事．ACM，IEEE 各会員．

石原 進 （正会員）

1994 年名古屋大学工学部電気工学科卒業．1999 年同大学大学院工学研究科博士後期課程修了．1998 年日本学術振興会特別研究員．1999 年静岡大学情報学部助手．2001 年同大学工学部助教授．2014 年カリフォルニア大学ロサンゼルス校客員研究員．現在，静岡大学学術院工学領域教授．博士（工学）．モバイルコンピューティング，モバイルアドホックネットワーク，センサネットワークに関する研究に従事．IEEE，ACM，電子情報通信学会各会員．本会シニア会員．