ネットワークシミュレータ MobiREAL を用いたモバイルアドホックネットワークシステムの開発環境

前田 久美子[†] 梅津 高朗[†] 山口 弘純[†]
安本 慶一[††] 東野 輝夫[†]

本論文では現実的なノードの行動を再現可能なワイヤレスネットワークシミュレータ MobiREAL を用いた実機・シミュレーション混在型の MANET アプリケーション開発支援環境を提案する．提案環境ではシミュレーションの進行を実時間に同期させ，実機上のアプリケーションと連携させることにより，実装固有の問題の検証を比較的大規模なネットワーク環境で行うことが可能である．また，シミュレーション中にノードの移動経路をインタラクティブに操作する機能により開発効率の向上を図っている．さらに MobiREAL で採用しているアプリケーション操作シナリオを利用し，実機でもシミュレーションと同じシナリオでアプリケーションを動作させることを可能とするインタフェースを提供することで，複数の実機を用いた評価における負荷の軽減も図っている．これらに加え，シミュレーションの実時間との同期精度を向上させるために，一部のタスクを時間的に分散して実行させる手法も提案し，実験により 100 ノードまでのシミュレーションならシミュレーション時刻の遅延の最大値を 35%〜45%程度に削減できるという結果を得た．

Development Environment for Mobile Ad-hoc Network Systems Using Network Simulator MobiREAL

Kumiko Maeda,[†] Takaaki Umedu,[†] Hirozumi Yamaguchi,[†]
Keiichi Yasumoto[††] and Teruo Higashino[†]

In this paper, we present new environment to support development of MANET applications using the network simulator MobiREAL that can reproduce realistic behavior of mobile nodes. In the proposed environment, we can synchronize the simulation clock with real time to achieve real-time simulation. By cooperating with a real application on real terminals, we can carry out large-scale simulation including the real terminals and can find the problems in real code that are difficult to find when we use virtual code on simulators. In the environment, we can interactively specify the movement of nodes during simulation so as to increase the efficiency of development. Also the environment enables us to operate the application automatically, in order to save the cost to conduct experiments with real terminals. In addition, we propose a method to reduce the delay of simulation progress to keep timeliness of real-time events. From the experimental results, we could reduce the maximum delay of events from real time by 35%–45% compared with ordinary simulation.

1. はじめに

近年の計算機システムのユビキタス化や無線技術の普及にともない，移動通信端末がアクセスポイントなどを用いずに自律的に分散・協調して通信するモバイルアドホックネットワーク（MANET）の研究がさかんに行われている．一般にシステムの開発プロセスにおいては，様々な環境におけるシステムの性能評価や挙動の正しさの検証が不可欠であるが，特に MANET 上のシステムの場合は，実環境において大規模な実験を行うことはコストなどの観点から現実的ではない．そのため MANET の性能評価にはネットワークシミュレータが用いられることが多い．

しかしながら，ネットワークシミュレータを用いた性能評価ではシステムの記述にシミュレータ用の言語を用いることから，シミュレータ上のシステムの動作と，実際のシステムの動作に相違が生じ，性能評価に影響を及ぼす可能性がある．特にアプリケーションの

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

設計を考えた場合，ユーザとのインタラクションなど通信以外の処理が多く存在し，またプログラム自体も複雑になりやすいことから，シミュレーションの段階で実際のアプリケーションになるべく近いコードを用いた性能評価を行うことが望ましい．

実機を用いた無線ネットワークのためのエミュレーション環境はいくつか提案されている[1)~4)]．文献 1) では，シミュレーションとエミュレーションを混在させ，エミュレーション用プロトコルのオーバヘッドを小さくすることで，スケーラビリティを考慮した環境 TWINE を提案している．また文献 2) では，物理ネットワークをシミュレートで補い，MAC 層でパケットをフィルタリングすることで，ノードを実際に移動させることなく，モビリティを考慮したエミュレートが可能な環境 MobiEmu を提供している．また，文献 3) では，ロボットを用いることで物理的に無線端末を移動させることが可能な環境 Mobile Emulab を提案している．また，文献 4) では，ns-2 が提供する仮想ネットワークと実ネットワークを接続する機能により，実ネットワーク部でアプリケーションを動作させ仮想ネットワークを利用して性能評価を行うことを可能としている．特に文献 1) により，無線ネットワークを介した動画のストリーミングにおいて，スループットなどのシミュレーションにより得られる統計的な情報と，人間が動画を実際に見ることで得られる画質などに対する感覚には差異がみられることが分かっており，実際のアプリケーションなどにより動作確認を行うことの重要性が述べられている．

本研究では我々が開発したワイヤレスネットワークシミュレータ MobiREAL[5)] を用いた MANET アプリケーション開発環境を提案する．提案する開発環境ではシミュレータ用のアプリケーションコードと，シミュレータとは独立したプログラムとして存在する実機コードとのハイブリッドシミュレーションを可能にしている．シミュレーションとエミュレーションの混在という点では提案環境は最も文献 1) に近いといえるが，MobiREAL による現実的なノードの行動・移動を利用し，かつエミュレーションが持つインタラクティブ性を生かした評価検証を可能にしたという点が既存研究にはない特徴である．以下具体的な説明を行う．

提案環境では，実機での開発を意識し，いくつかの典型的な実機の実行環境が提供する API になるべく近い通信用の汎用 API を実装する．現時点では UNIX のソケット通信ライブラリに似せたトランスポート層プロトコル（TCP，UDP）用の API を提供している．この API を介して送信されるパケットはシミュレータ上でその送信がシミュレートされ，適切な遅延や損失率のもとで配信される．実パケット送信がリアルタイムでシミュレートされることにより，実環境に近いネットワーク環境でのアプリケーションの実行検証を行うことができる．

また，提案する環境で用いるネットワークシミュレータ MobiREAL は特に人の行動の現実性に着目して開発されており，ネットワークシステムのシミュレーションとノードの行動（移動）のシミュレーションを連携させるフレームワークに基づき実装されている．この連携により，たとえば道路の混雑情報をノード間で交換し，その情報に従ってノードが混雑地点を迂回するために移動経路の変更を行うことや，特定のノードと遭遇したときにアプリケーションの実行を開始するなど，ノードが自身の周辺環境やネットワークシステムの振舞いに応じて行動を動的に変化させる挙動が再現可能である．従来，MobiREAL ではユーザの動的な行動を Condition Probability Event（CPE）モデルと呼ばれる確率を併用したルールの集合によりエージェント風に定義していたが，提案する開発環境では，この CPE モデルに基づくノードの行動をインタラクティブに実行するためのインタフェースや GUI を提供し，移動経路や速度に関するユーザの振舞いを外部入力として与えられるようにする．さらに，実機上のアプリケーションプログラムを CPE モデルに従って自動操作するためのインタフェースも提供する．このインタフェースに従い実機アプリケーションを実装することで，シミュレーション用のアプリケーション操作記述に従って実機上のアプリケーションも同様に操作することができ，シミュレーションコードと実機コードが同じシナリオで試験可能となる．これにより，シミュレーションでは得られない実装固有の問題の発見が容易になると考えられる．

また提案環境では実機とシミュレーションを連携させて動作させる必要があるため，シミュレーションの進行を実時間に同期させる機能（リアルタイムシミュレーション）が重要となってくる．この際，ある時刻にイベントが集中するなどして一時的に発生する遅延は，実機との同期を考えた場合その正確さに影響を与えるが，本研究ではその正確さを向上させるための 1 つのアプローチとして，一部の処理を時間的に分散させて実行することによりリアルタイム性を向上させる手法を提案する．5 台の実機を用いた実験により 100 ノードまでのシミュレーションなら実時刻からの遅延の最大値を通常のシミュレーションの 35%〜45%程度に抑えられることが分かった．

以上のように提案する開発環境ではMobiREALが提供する現実的なノードの行動・移動を実機の評価でも利用できる．特に，シミュレータ上の仮想ノードと実機上の実ノードを共通の操作シナリオで区別なく操作することが可能なインタフェースや，ノードモビリティをインタラクティブに操作する機能を提供することにより，既存のエミュレーション環境に比べ，より現実的なノードの行動モデルのもとでより効果的な評価実験を行うことが可能である．

2. MobiREAL

まず，本研究で提案する開発環境で使用する，我々が開発したMANET向けネットワークシミュレータMobiREAL[5]について，その特徴と概要を説明する．

MobiREALはMANETの性能評価において重要な要素である，ノードの移動の現実性に着目して開発されたシミュレータである．MANETではノードの行動とネットワークシステムの振舞いが互いに影響を与えながら状態を動的に変化させていく状況が存在する．このような動的相互関係をシミュレーションで再現するために，我々はネットワークシステムのシミュレーションとノード行動のシミュレーションが連携し，互いの状態を反映しながら全体のシミュレーションを進行するフレームワークを考案し，そのフレームワークに基づきMobiREALを実装している．さらに動的なノードの行動を容易に記述可能なモデルとして確率付きルールベースの行動記述モデルCondition Probability Event Model（CPEモデル）を提案している．本研究で提案する開発環境では，このCPEモデルを利用した実機コードの自動操作インタフェースやGUIを提供することで，実機を交えたシミュレーションにおいてもノードの行動の現実性を考慮したテスト環境を実現する．

MobiREALでは数千から数万ノード規模のシミュレーションを実現するため，複数台の計算機を用いた並列シミュレーション機能を提供している．並列シミュレーションによる効果はシミュレーション内容にもよるが，2台の計算機で1.5〜4.0倍程度の実行速度の向上率を達成している．

また，MobiREALシミュレータではシミュレーション結果を視覚化するアニメータを提供している．アニメータはWindows上で動作し，ノードやリンク，無線到達半径，パケット伝搬の様子などをアニメーションで表示できる．アニメータ実行時の静止画像と動画像はMobiREALのウェブページ[6]で公開している．提案環境ではシミュレーションの進行中に状態を表示できるようにこのアニメータを拡張し，さらにアニメータを通じてシミュレーション中にノードの移動をインタラクティブに操作できる機能を追加した．これにより，シミュレーション全体の状況を把握しながらノードの移動を変更したりアプリケーションを操作したりすることが可能となり，テストやデバッグの効率向上が期待できる．

3. 提案する開発環境の構成

提案する開発環境の概要を**図1**に示す．提案方式ではシミュレータ上に存在する従来の仮想ノードと，シミュレータとは独立したプログラムとして実行されている実機ノードが混在した環境でシミュレーションを行う．実機ノードは仮想ノードの1つにマッピングされる．実機ノードの通信部は実際のライブラリの代わりにシミュレータと連携するための専用APIを用い，このAPIを介してシミュレータと送受信するデータの情報をやりとりする．シミュレータとの連携用APIは現在UNIXのソケット通信ライブラリに似せたトランスポート層プロトコル（TCP，UDP）用のものを提供しており，接続先（実機のIPアドレスまたはシミュレータ内でのノードID，ブロードキャストアドレス）を指定してのTCP接続の確立やデータの送信，およびデータの受信などが可能である．将来的にはネットワーク層以上を実機上で動作させることができるよう，データリンク層API（仮想ネットワークデバイスインタフェース）を提供することも検討している．シミュレータと実機ノードとの連携はTCP通信により実現され，実機ノードに相当するプログラム（実機コード）が実行される計算機とシミュレータが実行される計算機はLANなどで接続されているとする．実機コードの動作環境はTCPソケット通信を利用できることが条件となる．

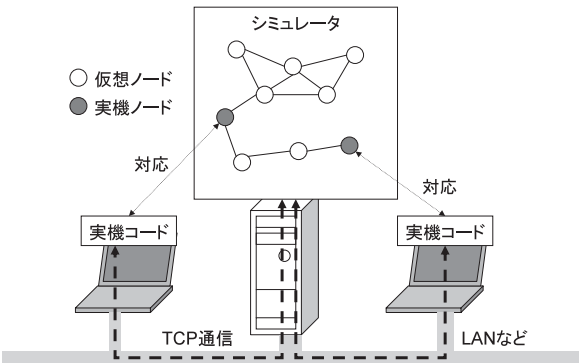


図1 提案環境の概要図
Fig. 1 Overview of proposed environment.

表 1 CPE モデルによるノードの行動記述例
Table 1 Example of node behavior description with CPE model.

	Condition	Prob.	Action
E1	GetNewAppOut() == "SHOP_A" アプリケーションから SHOP_A の情報を取得	0.30	PATH = shortest_path(P,Point(SHOP_A)) 目的地を SHOP_A に変更
E2	P == Point(SHOP_A) 頂点 SHOP_A に到着	0.10	InputApp("SHOP_A") アプリケーションに SHOP_A の情報を入力
E3	(P == Point(INT_1)) ∧ ((T%2min) < 1min) 交差点 "INT_1" で青 1 分，赤 1 分の信号を実現	1.00	stop(1min-(T%2min)); 信号が青になるまで待つ

※ P：ノードの現在地，PATH：予定移動経路，T：シミュレーション時刻

4. CPE と実機ノードの行動操作

本章では，CPE モデルの概要および，本論文で提案する CPE モデルを利用した実機の行動操作のためのインタフェースについて説明する．

4.1 CPE モデルの概要

MobiREAL で行動記述モデルとして採用している Condition Probability Event（CPE）モデルは，周囲の状況に応じたノードの行動の変化の規則を確率つきで記述するモデルである．CPE 記述例を**表 1** に示す．CPE 記述は，ノードのプリミティブな行動とその実行条件および実行確率の組である “ルール” の並びと位置や移動速度などのノードの状態を表す内部変数，および時刻などのシステムの状態を表す外部変数で構成され，条件を満たしたルールの行動が指定された確率で実行される．表 1 のシナリオでは近距離無線通信デバイスを装備した情報端末を保持する歩行者が店舗を訪れた際に入手した情報（たとえばセール情報）を，移動中に遭遇した他の歩行者に配布することで有用な情報の共有を行う MANET 上のネットワークシステムを想定しており，たとえばルール E1 では MANET を通して入手した情報により，目的地（移動経路）を変更するという行動を記述している．CPE 記述により変更できるノードの行動は，ルール E1 のような移動経路やルール E3 のような移動速度などのノードの位置情報に関係するもの，ルール E2 のようなアプリケーション操作に関係するものの 2 種類に分類できる．

4.2 実機用行動操作インタフェース

提案環境では，この CPE モデルをベースとした移動速度や移動経路の手動操作インタフェース，および CPE ルールに従い実機上のアプリケーションを自動で操作するためのインタフェースを提供する（**図 2**）．ユーザはアニメータを介してノードの移動経路を決定できる．また，CPE ルールにおいて確率で決定していた判定を GUI を介して外部からの入力として CPE に指定できるようにしている．

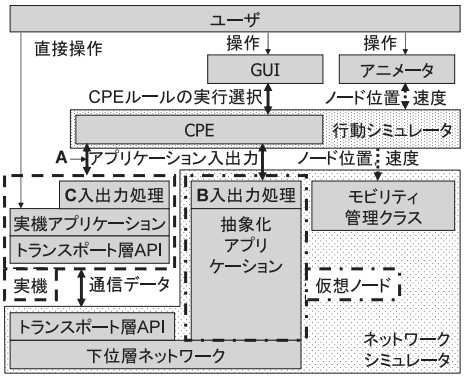


図 2 GUI・インタフェース概要
Fig. 2 Overview of GUI and interface.

次に，CPE に従った実機アプリケーションの自動操作インタフェースについて説明する．提案環境では，CPE ルールに従った実機アプリケーション操作手法を 2 通り用意している．1 つ目は仮想ノード用のアプリケーションコードと同様に，ソケット通信により行動シミュレータから出力される文字列形式のアプリケーション入力（図 2 矢印 A）を解析し，対応する関数などを実行する手法である．実機アプリケーションの記述言語が C や C++の場合など，実機の環境によっては仮想ノード用の入出力処理部（図 2 B）をほぼそのまま実機用の入出力処理部（図 2 C）に流用できるほか，簡単な入出力なら 1 入出力あたり数行〜十数行で実装可能である．

2 つ目はアプリケーション入力に対応するボタンやフォームなどの GUI を自動で操作する手法である．この手法は Java のバイトコード改変機能を持つ Javassist [7] を利用して実現する．このため，現状ではアプリケーションが Java で実装されている場合にのみ適用可能である．Javassist は，Java のバイトコードの詳細な知識を必要とせず，バイトコードの内容を変更するプログラムを書くことが可能なライブラリであり，クラス定義の変更やフィールドやメソッドの追加，メソッドの中身の変更が可能である．この Javassist を利用することにより，Java の標準 GUI クラスのボタ

ンやフォームを，入出力処理用ライブラリから操作できるように改造しておいたクラスと差し替えることで自動操作を実現する．ユーザは CPE ルールから生成される文字列形式のアプリケーション入力（図 2 矢印 A）がどのボタンやフォームに対応するかを別に指定し，その情報をもとに入出力処理用ライブラリ（図 2 C に相当）が生成される．この方法は実際のユーザの操作により近い状態で実機上のアプリケーションを実行できる利点がある．

5. リアルタイムシミュレーション

5.1 リアルタイムシミュレーションの実装方法

リアルタイムにシミュレーションを行う機能は，提案環境において重要な要素の 1 つである．リアルタイムシミュレーションではパケット送信などの各シミュレーションイベントを正確な時刻に実行することが，シミュレーションの正確さの観点から重要であると考えられる．しかし提案環境で使用しているネットワークシミュレータ MobiREAL は，実装の容易さや実行効率の観点から行動シミュレータとネットワークシミュレータがあらかじめ定義されたシミュレーション時間分の処理を終えるたびに同期して状態の更新を行うため，同期オーバヘッドによりイベント処理が実時間進行に対し遅延しやすくなる．したがって，シミュレーション全体を高速化するだけでなく，このような一時的遅延増大を抑制する工夫が必要となる．

本論文では，多少の遅延を許容でき，かつ処理に時間のかかるイベントを分割し，時間的に分散させて実行することにより，シミュレーションの一時的な遅延を抑える手法を提案する．分割実行が可能であるイベントとしては，行動シミュレータとネットワークシミュレータとの通信，およびそれにともなうノード情報の更新処理のほかに，トレースモビリティの読み込みや，統計データの収集，処理，出力などがあげられる．

提案するイベント分割実行手法の詳細を説明する前に，まずシミュレーション時刻を実時刻に同期させる手法について説明する．米 Georgia Institute of Technology で設計，開発されたネットワークシミュレータ GTNetS[8] をベースとして開発した MobiREAL は，離散イベント（Discrete Event）タイプのネットワークシミュレータである．Discrete Event Simulation（以下 DES）は，代表的なシミュレーション方式の 1 つであり，タイムスタンプ付きイベントを実行単位とするシミュレーションである．イベントは，「パケットの送信開始」などある瞬間の事象を表すオブジェクトであり，タイムスタンプはそのイベントが処理されるシミュレーション時刻を表す．DES ではタイムスタンプ順にキューにイベントが格納されており，先頭のイベントを取り出して処理を行い，その過程で新たに発生したイベントをさらにキューに格納する，という作業を繰り返すことでシミュレーションが進行する．

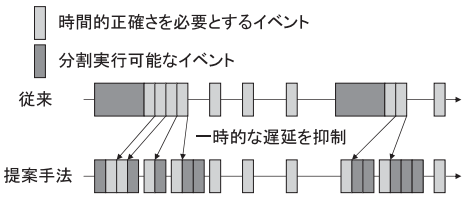


図 3 分割実行手法の概要
Fig. 3 Overview of event division and processing method.

この DES における実時刻との同期は以下のように行う．まずシミュレーション開始時にローカルマシンの実時刻 T_0 を取得し，以降は次に処理するイベントのシミュレーション時刻を S_{next}，現在の実時刻を T_{now} とすると，式 $T_{now} - T_0 \leq S_{next}$ を満たすまで待機してから次のイベントを処理する．提案手法では，この待機時間を用いて遅延の許容できるイベントを処理する．**図 3** に分割実行手法の概念図を示す．提案手法では，全イベントから遅延の許容できるイベント（分割実行イベント）をいくつか抜粋し，リアルタイム性が求められるその他のイベントとは別のキューを用いて管理，通常のイベントを優先して処理する．具体的には，分割実行イベントを最小処理単位に分割し，現在の待機時間がある一定時間（δ とする）以上であれば分割したイベントを 1 つ処理し，さらに実時刻を取得して待機時間を判定する．これを待機時間が δ に満たなくなるまで繰り返す．δ は最小処理単位の処理時間よりも長い必要があり，通常は十分余裕を持って見積もっておく．また，分割処理イベントの遅延が大きくなりすぎることを避けるため，便宜上分割処理イベントには処理時刻のデッドラインを設けておき，デッドラインを過ぎても処理されていない分割処理イベントが生じた場合はそれらを即座に処理する．分割処理イベントは，キューによりデッドライン順に管理する．

5.2 実　　験

本節では，リアルタイムシミュレーションが可能なシミュレーション規模の検証および，前節で述べた機構による効果の検証を行うために，シミュレーションに要した時間およびイベント処理時刻の実時刻に対する遅延の測定を行った．

実験に先立ち予備実験を行った結果，遅延が生じる最大の要因は行動シミュレータとの同期後のノード情報の更新処理であることが判明した．この処理を提

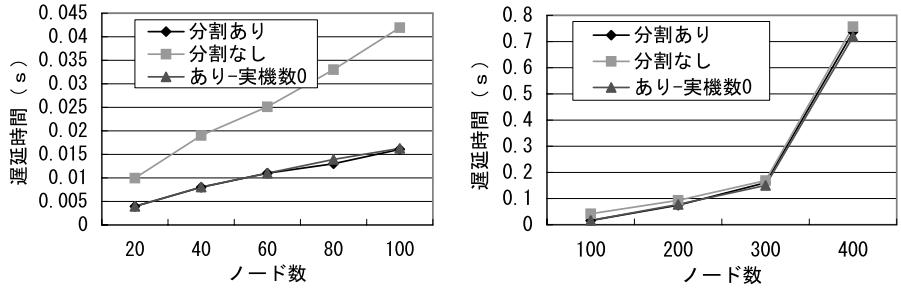


図 5 実時刻に対するシミュレーションの遅延
Fig. 5 Simulation delay to real time.

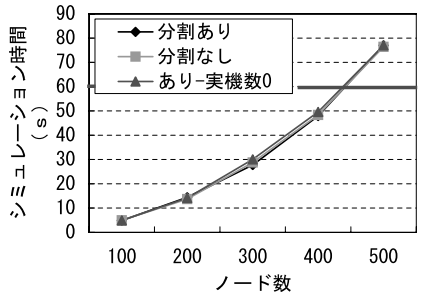


図 4 60 秒間のシミュレーションに要する時間
Fig. 4 Running time of 60 sec. simulation.

案手法を用いて分割実行させた場合（分割あり）と，従来どおり同期直後にまとめて実行した場合（分割なし）とで比較を行った．シミュレータは一般的な性能のデスクトップ PC （CPU Pentium4 3.40 GHz, メモリ 2 GB）上で，実機ノードはノート PC（CPU PentiumM 1.6 GHz，メモリ 1.5 GB）上で動作させ，並列シミュレーションを行わない状態で測定を行った．シミュレーションでは実機ノードを含む各ノードに 5〜10 秒間隔でランダムにシングルホップマルチキャストパケットを送信させた．特に明記しない場合は，実機ノード数は 5 とする．シミュレーション領域は 500 m×500 m，シミュレーション時間は 60 秒，MAC 層は IEEE 802.11 DCF + RTS/CTS を用い，各ノードの無線伝播距離は 100 m と設定した．ノードの移動モデルには Random Waypoint [9] を用いた．

図 4 にノード数 N を変化させた場合のシミュレーション時間（シミュレーション全体に要した時間）を示す．当然であるが，提案手法を用いるか否かにかかわらず，全体での処理量はほぼ同じとなるためほとんど差はみられないことが分かる．また，実機を 5 台用いている “分割あり” と，実機数が 0 台の場合の “あり–実機数 0” とを比較する限り，実機を用いることによる処理量の増加はほとんどないといえる．図 4 によると，500 ノードの場合は 60 秒のシミュレーションに 70 秒以上要しているため，今回の実験環境においては，400 ノード超の規模までリアルタイムシミュレーションを行えていることが分かる．

次に，**図 5** にノード数 N を変化させた場合の実時刻に対するシミュレーション時刻の遅延（イベントが本来処理されるべき時刻からの遅延）の最大値を示す．実機を用いることによる遅延の影響はほとんどみられない．またノード数が 100 まで（図 5 左）であれば，提案する手法を用いてノード情報の更新処理を分割実行させることで，提案手法を用いない場合に比べて最大遅延がおよそ 35%〜45%に減少していることが分かる．このようにある程度の遅延が許容されるイベントを分割して実行することにより，全体では同じシミュレーション時間でも，イベントごとのリアルタイム性を向上させられることが分かる．ただしノード数が多くなるとパケット送信数や 1 パケットあたりの受信ノード数の増加にともないイベント数が大幅に増加し，またイベントの種類によっては処理時間も長くなるため，δ 時間以上余裕のある待機時間が少なくなり分割実行による効果が得られにくくなると考えられる．このような遅延を解消するためには，シミュレーション自体を簡素化する，より高性能なマシンを用いる，並列シミュレーションを行う，などシミュレーションを高速化する工夫が必要となる．

なお，ノード情報の更新処理を分割実行することによりリアルタイム性は向上するが，一方で位置精度低下の問題が発生する．MobiREAL ではノードの位置情報が行動シミュレータからネットワークシミュレータに与えられる．本実験では行動シミュレータとネットワークシミュレータとの同期間隔を 1.0 秒としたため，その間はネットワークシミュレータが独自にノード移動を予測して補完し，次の同期時に補正する．しかし，提案手法ではノード情報の更新処理のデッドラインを同期後 1.0 秒後（次回の同期まで）としたため，ネットワークシミュレータの補完は最大 2.0 秒間分と

なり，その分位置精度が低下する．したがって，同期間隔やデッドラインなどのパラメータを適切に設定することにより，精度の悪化を抑えつつ，リアルタイムシミュレーションとしての時間的な正確さを向上させることが現在の課題である．同期間隔とノード座標誤差の関係に関する評価については文献 5) で行っている．

6. 事例適用による機能考察

本章では簡単な実験例を通して，インタラクティブな実機ノード操作機能の有用性について考察する．

この事例では，MANET 上で広告情報や災害などの緊急時における避難情報などを配布する位置依存データ配信システムのシミュレーションを行った．情報の配布は確率を用いたフラッディングベースの方式で行われ，情報配信元や近隣ノードから配布された情報を受信したノードは，その情報を隣接ノードに 50%の確率でほぼ即時に再配布する．

シミュレーション領域は 500 m×500 m，シミュレーション時間は 800 秒，情報配信元からの情報配信は 100 秒経過後から開始し，以降は 5 秒間隔で行う．MAC 層は IEEE 802.11 DCF + RTS/CTS を用い，各ノードの無線伝播距離は 100 m と設定した．モビリティモデルは文献 10) の実験で用いた，ノードが現実的な移動経路をたどりながら指定された密度分布を構成するモデルである Urban Pedestrian Flow（UPF）を使用し，ノード総数は約 400，ノードの移動速度は 1.1〜1.7 m/s とした．ノードの分布を**図 6** 右側に示す．UPF の特筆すべき特性として，シミュレーション中随時ノードが発生，消滅することでノードが次第に入れ替わっていくことがあげられるが，おおよそのノードの分布は時間が経過しても変化しない．

実験では情報配信元が図 6 左側に示される地図上の経路 A，B，C，D をたどって 2.0 m/s の速度で移動した場合についてそれぞれ情報を受信したノード数を測定した．各経路に従って移動した場合の，シミュレーション時間に対する情報受信ノード総数の推移を**図 7** に示す．このように経路によって情報受信ノード数にある程度の差がみられる．

この実験から分かるように，送信元の移動経路を変えるだけでその性能が大きく異なる場合が多い．たとえば最適な移動経路を探すことが目的である場合，設定を変えたシミュレーションを繰り返す必要があるが，提案環境ではこの情報配信元ノードを実機ノードとすることで，移動経路をインタラクティブに決定することができ，効果的な経路を効率良く発見することが可能となる．このように，インタラクティブにノードの移動やアプリケーションの操作を行うことで，パラメータの調整や機能のテストをよりスムーズに行うことができると考えられる．

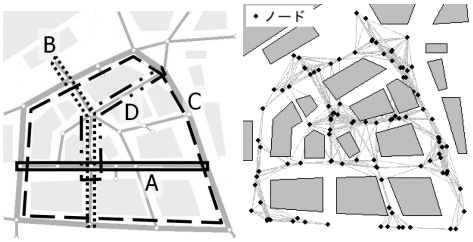


図 6 道路構造とノード分布
Fig. 6 Road structure and node distribution.

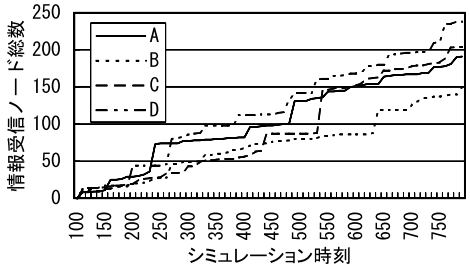


図 7 情報配布アプリケーション性能評価
Fig. 7 Performance of information diffusion system.

7. ま と め

本論文ではネットワークシミュレータ MobiREAL を用いた，MANET アプリケーション開発環境を提案した．提案環境では，実機のアプリケーションコードと，シミュレーションコードを混在させてシミュレーションを行うことができ，実装固有の問題の検証を比較的大規模なネットワーク上で行うことが可能である．さらに提案環境では，ノードの現実的な行動に焦点を当てて開発された MobiREAL の特徴を生かし，CPE というノード行動記述モデルをもとに実機コードに外部入力を与えるための GUI や，実機上のアプリケーションをシミュレータ上のアプリケーションと同様に自動制御するためのインタフェースを提供している．

下位層を実機上で実行するための機能の追加などの拡張を行い，また様々なアプリケーションの開発事例を通して，提案環境の有用性を実証することなどが今後の課題である．

謝辞 本論文の作成にあたり，様々なご意見，ならびにご助力をいただきました松下電器産業株式会社の中村敦司氏に深く感謝いたします．

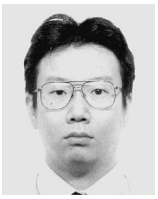
参 考 文 献

1) Zhou, J., Ji, Z. and Bagrodia, R.: TWINE: A Hybrid Emulation Testbed for Wireless Networks and Applications, *Proc. IEEE Infocom* (2006).
2) Zhang, Y. and Li, W.: An integrated environment for testing mobile ad-hoc networks, *Proc. ACM MobiHoc*, pp.104–111 (2002).
3) Johnson, D., Stack, T., Fish, R., Flickinger, D.M., Stoller, L., Ricci, R. and Lepreau, J.: Mobile Emulab: A Robotic Wireless and Sensor Network Testbed, *Proc. IEEE Infocom* (2006).
4) 宮地利幸，宇夫陽次郎，森島直人，篠田陽一：N*(NStar): ns-2 の real external interface の構想，情報処理学会研究報告（DPS），Vol.103, pp.113–118 (2001).
5) 前田久美子，小西一樹，佐藤和基，山口弘純，安本慶一，東野輝夫：現実的なシミュレーションシナリオが記述可能な無線ネットワークシミュレータ MobiREAL，情報処理学会論文誌，Vol.46, No.2, pp.405–414 (2006).
6) MobiREAL Simulator Web Page. http://www.mobireal.net/
7) Shigeru, C. and Muga, N.: An Easy-to-Use Toolkit for Efficient Java Bytecode Translators, *Proc. 2nd Int. Conf. on Generative Programming and Component Engineering* (*GPCE '03*), pp.364–376 (2003).
8) Riley, G.F.: The Georgia Tech Network Simulator, *Proc. ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pp.5–12 (2003).
9) Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.-C. and Jetcheva, J.: A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, *Proc. ACM/IEEE MobiCom*, pp.85–97 (1998).
10) Maeda, K., Sato, K., Konishi, K., Yamasaki, A., Uchiyama, A., Yamaguchi, H., Yasumoto, K. and Higashino, T.: Getting Urban Pedestrian Flow from Simple Observation: Realistic Mobility Generation in Wireless Network Simulation, *Proc. 8th ACM/IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (*MSWiM*), pp.151–158 (2005).

(平成 18 年 11 月 2 日受付)
(平成 19 年 4 月 6 日採録)

前田久美子（学生会員）

平成 18 年大阪大学大学院情報科学研究科情報ネットワーク学専攻博士前期課程修了．同年同大学院博士後期課程進学．アドホックネットワークに関する研究に従事．



梅津 高朗（正会員）

平成 13 年大阪大学大学院基礎工学研究科情報数理系専攻博士前期課程修了．同年同大学院博士後期課程進学．平成 14 年同大学院博士後期課程退学後，同大学院情報科学研究科助手．平成 19 年より同研究科助教．博士（情報科学）．アドホックネットワーク用ミドルウェアや開発環境の研究に従事．

山口 弘純（正会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院基礎工学研究科博士後期課程修了．同年オタワ大学客員研究員．平成 11 年大阪大学大学院基礎工学研究科助手．平成 14 年同大学院情報科学研究科助手．平成 19 年より同研究科准教授．博士（工学）．分散システムや通信プロトコルの設計および実装に関する研究に従事．IEEE，電子情報通信学会各会員．

安本 慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院基礎工学研究科博士後期課程退学後，滋賀大学経済学部助手．平成 9 年モントリオール大学客員研究員．平成 14 年奈良先端科学技術大学院大学情報科学研究科助教授．平成 19 年より同研究科准教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS，ACM 各会員．

東野　輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院基礎工学研究科博士課程修了．同年同大学助手．平成 2 年，6 年モントリオール大学客員研究員．現在，大阪大学大学院情報科学研究科教授．博士（工学）．分散システム，通信プロトコル，モバイルコンピューティング等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．