情報処理学会論文誌 Vol.59 No.10 1814–1826 (Oct. 2018)

仮想マシンによる大規模アドホックネットワークシミュレーション環境構築法

小比賀 亮仁[1,2,a)] 篠田 陽一[2] 岩井 孝法[1] 里田 浩三[1]

受付日 2017年12月22日, 採録日 2018年7月10日

概要：本稿では，仮想マシンによる大規模アドホックネットワークシミュレーション環境構築法を提案する．近年，これまで通信ネットワークとは関連の薄かった家電製品やその他のセンサがネットワークにつながりはじめ，それらが強く我々の社会生活に影響を及ぼしている．このような複雑な社会環境の理解には，様々なシミュレータを取り込んだ大規模社会シミュレーション環境の実現が望まれている．その具体例として，我々は仮想マシンを用いたアドホックネットワークシミュレーション環境の構築を考えた．仮想マシンを用いたアドホックネットワークシミュレーションは，仮想マシンとして具現化した車・人などを仮想空間上に配置し，それらが互いの位置を確認しつつアドホックネットワークを形成する．仮想マシンを利用することによって，仮想マシン内で様々な実アプリケーションを実行することが可能になり，より現実世界に近いシミュレーション環境を実現できる．しかしながら，仮想マシンを用いてアドホックネットワークのシミュレーションを実施する場合は，どの仮想マシンどうしが通信路を形成するかということが分からない状態で仮想マシンを配置することになるので，仮想マシンどうしが複数の物理マシンにまたがって通信路を形成し，物理ネットワークの通信輻輳を引き起こすことになり，シミュレーション環境を大規模化することが困難であった．本稿では，アドホックネットワークが形成する経路情報の局所性を活かして，地理的に近い場所にいる仮想マシンを同じ物理マシンに配置するという手法を用いることによって，物理ネットワーク上に形成される通信路を削減し，通信輻輳を低減させる仮想マシンの配置手法を提案する．また，実環境と同じ環境で仮想マシン内のアプリケーションを実行するために，シミュレーションフレームワークとして必要なコンポーネントとアーキテクチャを提示する．我々の研究によって，数千台の仮想マシンが動作する大規模アドホックネットワークシミュレーションを実現することが可能となる．

キーワード：仮想マシン，コンピュータシミュレーション，アドホックネットワーク

Creating Massive and Complex Ad Hoc Network Simulation Environment Based on Virtual Machine Technique

Akihito Kohiga[1,2,a)] Yoichi Shinoda[2] Takanori Iwai[1] Kozo Satoda[1]

Received: December 22, 2017, Accepted: July 10, 2018

Abstract: We suggest a mechanism to create a massive and complex ad hoc network simulation environment based on virtual machine technique. In the IoT (Internet of things) era, many IT devices are connected to the Internet and it makes massive and complex social environment. Therefore, massive and complex social simulation environment is needed to understand and estimate the impact how IoT drastically changes the human activities. However, many social simulation tools are closed to simulate human activity only. It is expected that a lot of network simulation tools and social simulation tools are merged. We considered massive and complex ad hoc network simulation environment as one of the embodiments such complex social simulation environment, especially ad hoc network simulation based on virtual machine environment. Ad hoc network simulation based on virtual machine technique can execute actual applications on a simulated network, which reduces costs for making simulation tools, gives us good understanding the phenomenon simulation makes, and so on. However, it is difficult to apply existing cloud management system because an unknown network topology created by virtual machines makes a partial network congestion of a physical network. Therefore, we suggest a novel mechanism for virtual machine deployment. Suggested mechanism executes virtual machines on one physical machine, which are "geographically" close to each other and when a virtual machine moves to another area, live migration is executed. We evaluated how much our mechanism reduces the physical network congestion and it is revealed that virtual machine deployment based on "geographical information" is feasible for ad hoc network simulation. We also create "external environment" of virtual machines to be resemble to the real world. Our suggestion embodies a massive and complex ad hoc network simulation on which thousands of virtual machines are executed.

Keywords: virtual machine, adhoc network, network simulation, scheduling

1. 背景

我々は StarBED [1] において，複雑な社会環境をコンピュータ上でシミュレートする大規模社会シミュレーションに関する研究を行っている．

文献 [2] でも述べられているように，社会シミュレーションは，人をエージェントとして表現し，その相互作用によって引き起こされる様々な社会現象（噂の伝搬経路や社会的なグループの形成過程）の理解や，新たな社会システムの導入効果（避難誘導システムによる避難にかかる時間の短縮効果など）を計測するという用途に利用されてきた．

一方，これまで通信ネットワークとは関連の薄かった家電製品やその他のセンサがネットワークにつながりはじめ，それらが強く我々の社会生活に影響を及ぼしはじめている．ネットワーク技術に触れる機会が飛躍的に増えた今，ネットワーク技術と人との関係を詳細に検証できる基盤が必要不可欠になりつつある．

そこで，我々は，これまで StarBED において培ってきた数々のネットワーク技術と社会シミュレーションを融合し，ネットワークと人との関係を詳細に検証できる新たな基盤の実現を考えた．これを我々は大規模社会シミュレーション環境と呼称する．

大規模社会シミュレーション環境は，現実世界に現れる人・モノがネットワークに接続される IoT の時代において，それらの相互接続が，実社会にどのような変化を及ぼすのかということを統合的に検証することを目的としている．そのため，大規模社会シミュレーションは，PC だけではなく携帯電話やセンサなど身近にあるデバイスと，それらをつなぐ温度場や電磁場まで，様々なシミュレータを取り込み，複雑な社会環境の再現を目指している．

大規模社会シミュレーション環境の具体例として，最初に，我々は人や車の移動により作られる現実的な無線環境の再現と，それによるアドホックネットワークの形成シミュレーションの実現を考えた．近年，南海トラフを震源地とする大規模な地震の発生確率が高まっているといわれている．文献 [3] によると，南海トラフ地震が発生すると，固定電話・携帯電話は約 1 割程度しか通話できなくなると予想されている．よって，携帯電話網の代替手段としてのアドホックネットワークを利用することができれば，災害に強い情報インフラ構築に役立つものと考えている．

また，我々が実現を目指すアドホックネットワークの形成シミュレーションは，**シミュレータ上で実際のアプリケーションを実行できること**が必要条件になると考えている．文献 [2] で述べられているように，シミュレーションによる新たな社会システムの導入効果を検証するためには当事者の参加が必要不可欠な要素になる．シミュレータ上で通話アプリケーションを実行して，実際にその通話品質を体感することで，実用性に関する高い立証効果を示し，今後の災害対策提言に役立てることを狙っている．

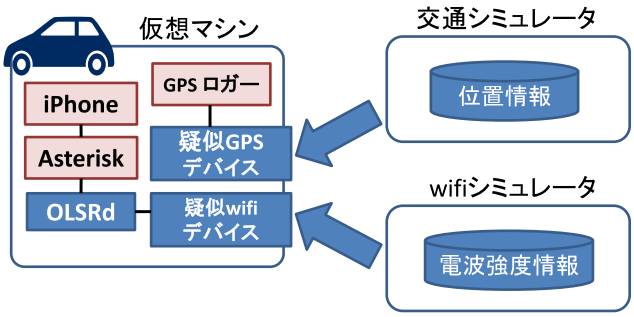

図 1 仮想マシンを使ったアドホックシミュレーションのシステム構成例

Fig. 1 A system configuration for adhoc network simulation based on virtual machine technique.

そこで，実用性の高いアドホックネットワークシミュレーションの構成方法として，我々は仮想マシンに着目した．仮想マシンを使ったアドホックネットワークシミュレーションは，仮想マシンとして具現化した車・人などを仮想空間上に配置し，それらが互いの位置を確認しつつアドホックネットワークを形成する．仮想マシンを使うことによって，当該仮想マシン上で実アプリケーションを動作させて，より現実に近い評価が可能となる．

図 1 は仮想マシンを使ったアドホックネットワークシミュレーションのシステム構成例を示している．図中，Asterisk [4] は SIP サーバプログラム（電話交換機に相当）である．また，GPS ロガーは，GPS デバイスから得られる位置情報を記録するためのアプリケーションである．OLSRd（Optimized Link State Routing daemon）は，アドホックルーティング情報を生成するためのデーモンプログラムを指している．疑似 GPS デバイス，および疑似 wifi デバイスは，GPS や wifi デバイスとして利用されるデバイスドライバを指している．これらのデバイスは，仮想マシンの外で実行されている交通シミュレータや wifi シミュレータなどから，それぞれ位置情報や他の仮想マシンと通信する場合の電波強度情報を取得する．電波の強弱は，個々の仮想マシンが持つ位置情報によって異なる．見かけ上（交通シミュレータによって得られるそれぞれの車両の時刻ごとの位置上），離れている仮想マシン間で通信する場合は，電波強度が弱くなるので，ネットワークパケットを喪失しやすくなり，近くにいる仮想マシン間で通信する場合は，電波強度が強くなるので，ネットワークパケットの喪失が少なくなる．OLSRd は，パケットの喪失状況をもとにルーティング情報を形成する．このアドホックネッ

[1] 日本電気株式会社
NEC Corporation, Kawasaki, Kanagawa 211–8666, Japan
[2] 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923–1292, Japan
[a)] a-kohiga@cp.jp.nec.com

トワークシミュレーション環境で実際に通話する場合は，iPhone などの端末を Asterisk に接続して利用することが可能である．

このようなシミュレーション環境において，数千から数万の仮想マシンを参加させて，大規模なアドホックネットワークを形成することによって，広域の通話網を形成する．実際に携帯電話などから通話し，その通話品質を体感することによって，広域の災害に対しても，実用的なアドホックネットワークが形成できるということを立証することができ，今後の災害対策提言に役立つものと考えている．

2. 関連研究

2.1 ネットワークシミュレーションツール

従来，ネットワークシミュレーションは，NS3 [5] や OMNET++ [6] などのネットワークシミュレーションツールが用いられてきた．NS3 や OMNET++ は，キューイングシステムを C++ で記述するシミュレーションツールである．これらのシミュレーションツールは，キューイングシステムの出力として，TCP/IP などの実際のプロトコルに従った「疑似パケット」を生成することができる．

一方，これらのネットワークシミュレーションツールを用いると次のような問題がある．1. ツール独自の仕様を覚えて，シミュレーションネットワークを一から作る必要がある．2. 当該シミュレーションネットワーク上で，実際のアプリケーションを評価できない．以上の理由から，近年では仮想マシンを使ったネットワークシミュレーションが注目されつつある．

2.2 仮想マシンによるネットワークシミュレーション

Cloonix [7] や Netkit [8] は，QemuKVM ベースのネットワークシミュレータである．Core [9] は，Linux Container ベースのネットワークシミュレータである．いずれのツールも仮想マシンをルータや計算機と見立て，GUI を使ってそれらのつながりを入力したり，個々の計算機にログインしたりすることができる．

仮想マシンを使ったネットワークシミュレーションの研究は「時刻同期」が問題となる．人の流れや交通渋滞などを取り扱うような離散シミュレーションでは，エンティティ（人や車など）のすべてが時間的に同期のとれている状態でシミュレーションを進めなければならない．同期がとれていない状態でシミュレーションが進められると，本来，人が混雑するはずの場所で混雑が起こらない，車が渋滞するはずの場所で渋滞が起きないなど，シミュレーション結果の信憑性が失われてしまう．

「時刻同期」を取り扱った研究は，「実時間に依存した時刻同期方法」と「実時間から完全に独立した時刻同期方法」の 2 種類が存在する．文献 [10], [11], [12], [13] は，実時間に依存した時刻同期方法である．文献 [14] は，実時間から完全に独立した「仮想時間」を使った時刻同期方法である．

2.3 無線エミュレーションツール

我々の想定する環境は，車載器や携帯電話などを用いて災害に強いアドホックネットワークを構成することが目的なので，建物や妨害電波の発信元となる機器の有無など，仮想マシンによって具現化されるシミュレーション対象の外部の環境を考慮することが重要となってくる．

このような無線環境をエミュレーションするツールとしては，qomet [15] があげられる．qomet は，移動体の位置，移動速度など，あらかじめ作られたシナリオに基づいて，移動体と遮蔽物との位置関係から，各移動体間の通信遅延と帯域を推定して，時刻ごとに計算された通信遅延と帯域をネットワークインタフェースに設定することによって，疑似的な無線通信を実現している．

3. 課題

2.2 節および 2.3 節に示した既存技術を組み合わせると，仮想マシンによるネットワークシミュレーション環境を作ることが可能である．しかしながら，ここでの問題は，シミュレーション環境のスケーラビリティにある．

仮想マシンを使ったアドホックネットワークシミュレーションは，仮想マシン間で実際のパケット通信を行う．仮想マシン間の通信が 1 つの物理マシンに閉じている場合は，仮想マシン間の通信は共有メモリを用いた通信に置き換えられるので，通常のネットワークデバイスと比較して非常に大きな通信帯域を確保できる．しかしながら，仮想マシン間の通信が複数の物理マシンをまたぐ場合は，実際の物理ネットワークを介してパケットの送受信が行われるため，仮想マシン間通信の通信帯域が物理ネットワークの制約を受けてしまう．我々が提供するシミュレータのシミュレーション表現能力が物理的な制約を受けないためにもこのような状況は可能な限り防がなくてはならない．

しかしながら，アドホックネットワークシミュレーションのように，頻繁に通信相手が変わっていくような環境下において，ネットワークリソースの利用を最小化する仮想マシンの動的な負荷分散手法は存在しない．文献 [16] に示す従来の仮想マシン配置手法は，仮想マシンが利用するリソース量を測定しつつ，利用されるリソース量が物理マシン間で均一になるように仮想マシンのライブマイグレーション機能などを使って負荷分散を実施する．このような既存手法は CPU やメモリ，ネットワークなどのリソースの利用傾向が各仮想マシンに特徴的であり，頻繁に変わらないという前提条件のもとで，リソースパッキングの問題を解決している．一方，リソースの利用傾向が頻繁に変化するようであれば，そのつどライブマイグレーションを繰り返すことになるので，仮想マシンの移動処理によって，シミュレーション環境の CPU 時間やネットワーク帯域を

圧迫し，シミュレータとしての実用性を失う．仮想マシンを用いてアドホックネットワークのシミュレーションを実施する場合は，どの仮想マシンどうしが通信路を形成するということが分からない状態で仮想マシンを配置することになるので，仮想マシンどうしが複数の物理マシンにまたがって通信路を形成し，物理ネットワークの通信輻輳を引き起こすことになり，シミュレーション環境を大規模化することが困難である．

4. 解決方法

上記の課題を解決するために，我々はボロノイ分割による区画管理と物理マシンの割当てを基本とした大規模アドホックネットワークシミュレーション環境構築法を提案する．

4.1 ボロノイ分割による区画管理

通常，アドホックネットワークを形成する際は，移動体（我々のシミュレーションでは仮想マシンを指す）の近くにいる他の移動体と通信して，その帯域や遅延を計測しながら，ルーティング情報を作成していく．よって，近くに存在する移動体とは通信が頻繁に発生する．したがって，大規模アドホックネットワークシミュレーション環境構築法では，**近くにいる移動体どうしが頻繁に通信する**という特性（以下，通信の局所性と呼称する）を生かした仮想マシンの配置計画を考える．通信の局所性を高めるための仮想マシンの配置計画として，我々は 3 つのことを考えた．

1. 地理的な距離に基づいた仮想マシンの配置方法
2. ボロノイ分割による地理空間の分割と物理マシンへの割当て
3. シミュレーションクラウドの負荷分散手法

以下，それぞれについて説明する．

4.2 地理的な距離に基づいた仮想マシンの配置方法

ラウンドロビンなど，通常のスケジューリング手法を使って，仮想マシンを物理マシンに詰め込んでいくと，課題でも述べたように物理マシン間で通信輻輳を起こしやすくなる．よって，我々はあらかじめ，シミュレーションで用いられる 2 次元空間を複数の区画に分割し，それぞれの区画に対して，物理マシンを割り当てて，当該区画上で動作する移動体（車・電車など）を仮想マシンとして実行することとした．これによって地理上で近い移動体は同じ物理マシンに配置される可能性が高くなる（**図 2**）．

移動体がシミュレーションを進める中で区画をまたいで移動した場合は，仮想マシンのライブマイグレーションを使って当該仮想マシンを移動先の区画で利用されている物理マシンへ移動させる．そうすると，区画ごとに混雑する場所とそうでない場所ができる．この区画ごとの負荷の不均衡を解消する手段に関しては，4.4 節で説明する．

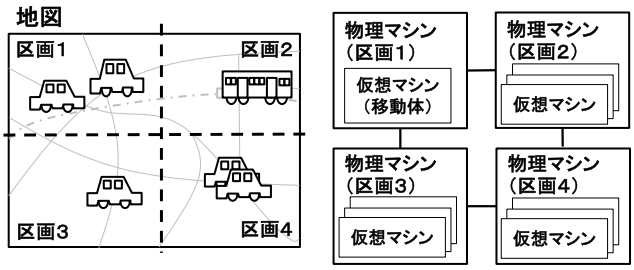

図 2 地理空間を考慮した仮想マシンの配置方法

Fig. 2 A deployment example of virtual machines. Each physical machine takes charge of a certain geographical area.

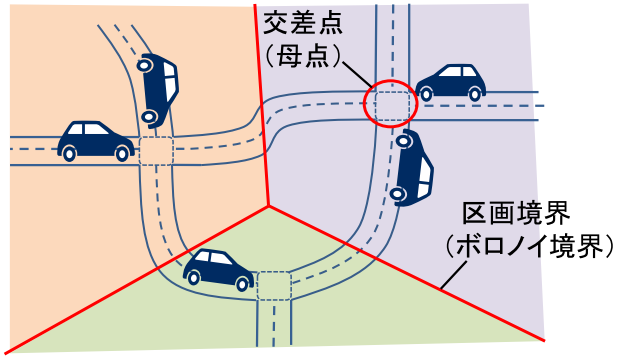

図 3 ボロノイ分割

Fig. 3 Dividing an area with voronoi diagram.

4.3 ボロノイ分割による地理空間の分割と物理マシンへの割当て

道路の交差点付近は，車が溜まりやすい場所であり，各移動体の間で通信が頻繁に発生することが予想される．図 2 と同じように 2 次元空間を一定の距離で分割して物理マシンに割り当てると，区画外縁部に交差点が現れる可能性がある．区画外縁部に交差点が配置されると，交差点付近で頻繁に通信する移動体が複数の物理マシンに分割して配置されることになる．頻繁に通信する移動体が複数の物理マシンに分割して配置されると，物理マシン間の通信帯域を圧迫してしまう．

このような状況を可能な限り少なくするため，ボロノイ分割による区画分割を取り入れる．ボロノイ分割とは，2 次元空間上に，複数の点が存在する場合に，点（ボロノイ分割では母点と呼ぶ）と点の垂直二等分線をもって区画の境界とする空間の分割方法である．分割された各領域をボロノイ領域と呼ぶ．ボロノイ分割における点を交差点と考えると，必ず交差点がボロノイ領域の中心に存在するようになるので，交差点が区画の外縁部に位置することがなく交差点付近に集中する仮想マシン間の通信を単体の物理マシン内で閉じることができる（**図 3**）．

単純にボロノイ領域のそれぞれに物理マシンに割り当てると交差点の数が増えれば増えるほど物理マシンが必要となるので，実際は，いくつかのボロノイ領域を束ねて 1 つの物理マシンに割り当てることになる．以下では，ボロノ

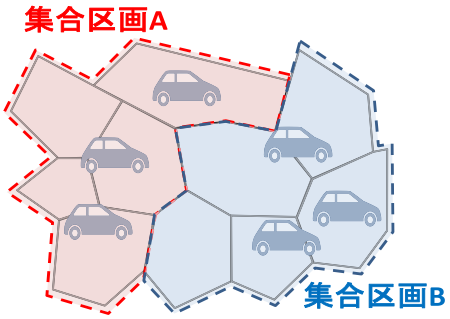

図 4 区画分割
Fig. 4 Divided areas.

イ分割によって分けられた領域を「区画」と呼び，物理マシンを割り当てるために束ねられた区画の集合を「集合区画」と呼ぶ．

4.4 シミュレーションクラウドの負荷分散手法

4.2 節でも述べたとおり，移動体が集合区画をまたいで移動すると，集合区画ごとに混雑する場所とそうでない場所ができる．この集合区画ごとの負荷の不均衡を解消する手段は，主に次の 3 つとなる．

1. ある物理マシンに割り当てられている集合区画を分割して，新しい物理マシンを追加する．
2. ある集合区画に存在する移動体（当該集合区画を担当する物理マシン上で実行される仮想マシン）が少なくなった，もしくはいなくなった場合は，当該集合区画を隣接する集合区画と合併する．
3. 物理マシンに割り当てられている集合区画を部分的に変更する．

本節ではこれらについて詳しく説明する．

(1) 集合区画の分割と物理マシンの追加

集合区画の分割とは，ある集合区画を担当する物理マシンの負荷が上昇して閾値を超えた場合に，当該集合区画を分割して，分割した片方の集合区画に物理マシンを追加する操作を指している．仮想マシンを用いた大規模なシミュレーションにおいては，物理マシンをまたいで仮想マシン間の通信が頻繁に起こらないように集合区画を分割するには，面積が半分になるように集合区画を分割するのではなく，**移動体の集合を 2 つに等分割し，かつ分割した 2 つの集合間での通信が最小となる集合**で集合区画を分割する必要がある（**図 4**）．そのため，集合区画の分割は次のような処理手順から構成される．

1. 分割対象となる集合区画に存在する移動体の集合を 2 つに分割する．
2. 1 で分割した 2 つの集合を基に，それぞれの移動体が所属する区画を 2 つの新しい集合区画として定義する．
3. ある区画に所属する移動体群が，1 で分割した 2 つの集合のそれぞれに所属する場合（ある区画に 2 つ以上の移動体が存在し，お互いが別の集合に属している場合），当該区画に存在する移動体群がどちらの集合に多く所属しているかを判定して，移動体群が多く所属している集合区画へ当該区画を割り当てる．
4. 2 つの新しい区画のそれぞれに物理マシンを割り当てる．この 2 台の物理マシンのうち 1 台は集合区画分割前に利用していた物理マシンである．
5. 2 つの新しい集合区画に所属する移動体のうち，新たに割り当てられた物理マシンで実行されるべき仮想マシンはライブマイグレーションを用いて移動する．

(2) 集合区画の合併と物理マシンの削除

集合区画の合併とは，ある集合区画を担当する物理マシン上で実行されている仮想マシンの数が減って，閾値を下回った場合に，当該集合区画に存在する区画の所属を周辺の集合区画へ移動させることを指している．

集合区画の合併では，区画を周辺の集合区画に移管するため，周辺区画に対する知識が必要となる．本稿では，ある区画の「周辺区画」を把握するために，「ドロネー三角形分割」を利用する．ドロネー三角形分割とは，ボロノイ領域の母点を，領域を隣接する他の母点と線で結び，母点を頂点とする三角形の集合として，2 次元空間を分割するような領域の分割手法を指している．これによって，ある交差点（母点）に隣接する他の交差点を把握することが可能となる．合併対象となる集合区画内に存在する区画に隣接する区画をすべて探索すると，当該集合区画に隣接する周辺区画（集合区画）が分かる．これを基に，最も近い距離に存在する周辺区画（集合区画）へ区画を移管する．集合区画の合併は次のような処理手順となる．

1. 合併対象となる集合区画に存在するすべての区画について，最も近い周辺区画（集合区画）を探索する．
2. 合併対象となる集合区画の存在するすべての区画について，最も近い周辺区画（集合区画）へ所属を移動させる．
3. 合併対象となる集合区画内の区画に移動体（仮想マシン）がまだ残っている場合は，当該区画の移動先となる集合区画（物理マシン）に，当該移動体を移動（ライブマイグレーション）する．
4. 最後に合併対象となる集合区画に割り当てられていた物理マシンを開放する．

(3) 区画割当ての部分的な変更

区画割当ての部分的な変更とは，隣接する 2 つの集合区画について，負荷の高い集合区画から負荷の軽い集合区画へ部分的に区画管理を移すことを指す．区画割当て変更は，任意のタイミングで実行可能だが，今回は，ある移動体が集合区画をまたいで移動する時点で実行することとした．また，負荷の高さを表す基準は，当該集合区画に割り当てられた物理マシン上で実行される仮想マシンの数を用いている．

図 5 は，2 つの隣接する集合区画（図中，集合区画 A

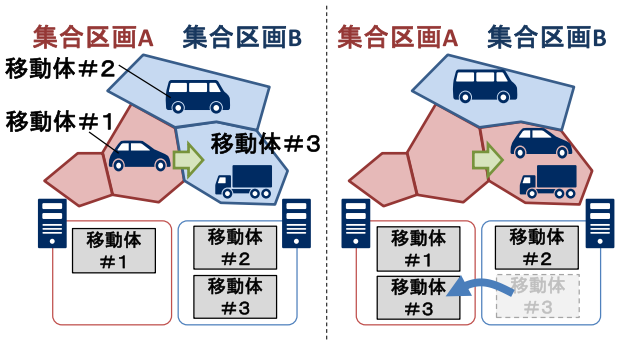

図 5 区画割当ての部分的な変更
Fig. 5 Partial change of area management.

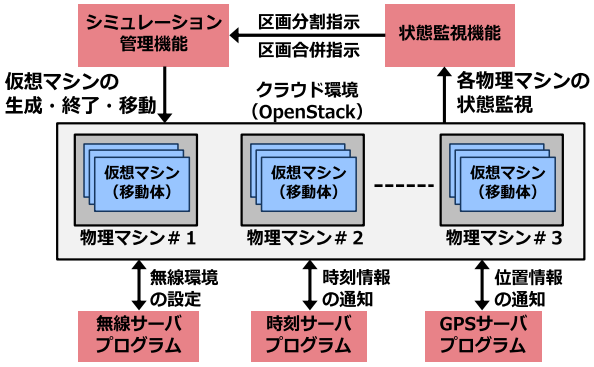

図 6 システムアーキテクチャ
Fig. 6 System architecture of this simulation system.

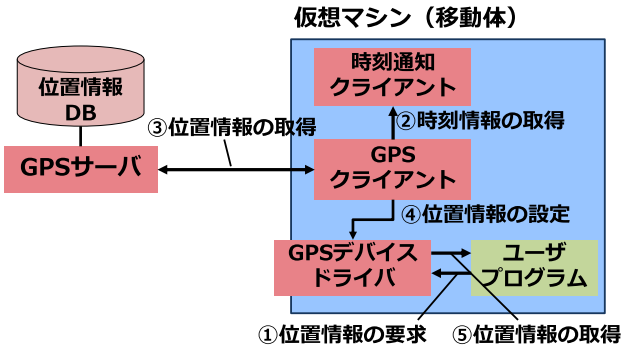

図 7 GPS サーバおよびクライアントプログラム構成
Fig. 7 Function layout of GPS emulation server and client program.

および集合区画 B）の区画割当て変更例を示している．図は，移動体#1 が集合区画 B へ移動する例を示している．図 5 の破線左図が移動前，破線右図が移動後の状態を示している．

区画割当ての変更処理は移動体#1 が集合区画 B へ移動する際に実施される．区画割当ての変更処理では，移動体#1 が新しい区画へ移動した状態における 2 つの集合区画の負荷を比較する．負荷とは，各区画に存在する移動体（仮想マシン）の数を指している．図を見ると，単純に移動体#1 が集合区画 B へ移動した場合は，集合区画 A には移動体が 0 台，集合区画 B には移動体が 3 台という状態になる．一方，移動体#1 の移動先となる区画を集合区画 A へ移管した場合（図中右側の状態）では，集合区画 A には移動体が 2 台，集合区画 B には移動体が 1 台という状態になる．よって，このような場合には，移動体#1 の移動先となる区画を集合区画 A に移管し，移動体#3 を集合区画 A に割り当てられている物理マシンへ移動することになる．

区画割当ての部分的な変更は，隣接する 2 つの区画の負荷の不均衡がそれほど大きくない場合に適用される．ある集合区画の負荷が大きくなった場合は，区画分割と物理マシンの追加が必要となる．

本節では，シミュレーションクラウドの負荷分散手法について，最も基本的な 3 つの操作について説明した．区画割当ての部分的な変更と集合区画の分割・合併を実行するタイミングによって，物理マシンの利用効率に差が出る．どのタイミングでこれらの操作を実行すると，最も効果的に区画を管理することができるかは今後の課題となる．

4.5 移動体としての仮想マシンの外部環境を備えるシミュレーションアーキテクチャ

図 6 にシステム全体の概要図を示す．シミュレーション管理機能は，シミュレーションの実行を担当する．状態監視機能は，クラウド環境内の物理マシン上で動作する仮想マシンの台数を監視して，仮想マシンの台数が閾値を超えた場合は，シミュレーション管理機能に対して，区画分割の指示を送信する．本シミュレーションでは，時刻・GPS・無線の 3 つの環境コンポーネントを用意した．これらの環境コンポーネントは，仮想マシン上のデバイスドライバに接続されており，仮想マシン上のアプリケーションは，実環境とシミュレーション環境の違いを意識しなくてもよい．

4.6 環境コンポーネント

環境コンポーネントは，各仮想マシンに対して GPS や時刻などの仮想マシンの外部の環境を提供する．これにより，シミュレーションの一部として実アプリケーションを実行できる．

(1) 時刻通知

時刻サーバプログラムは各仮想マシンに対してシミュレーション時刻を提供する．シミュレーション時刻は，クラウド環境に対してブロードキャストされ，仮想マシン上で実行される環境コンポーネントのクライアント部分で格納される．シミュレーション時刻を必要とする GPS や無線エミュレーションなどは，この時刻通知クライアントからシミュレーション時刻を取得する．

(2) GPS

GPS 環境コンポーネントは，各仮想マシンに対して位置情報を提供する．図 7 に GPS 環境コンポーネントの全体像を示す．図に示すとおり，GPS 環境コンポーネントは，

GPS サーバプログラムと GPS クライアントプログラム，GPS デバイスドライバから構成され，GPS サーバプログラムは位置情報 DB を持つ．ユーザプログラムが要求する位置情報は，GPS 環境コンポーネントにアクセスする疑似 GPS デバイスドライバを経由して取得される．位置情報 DB には，時刻ごとの各移動体の位置情報が格納されており，GPS クライアントは，時刻通知クライアントから得たシミュレーション時刻を GPS サーバに送信して当該時刻の移動体の位置情報を取得する．

(3) 無線エミュレーション

無線エミュレーション環境コンポーネントは，各仮想マシンに対して疑似的な無線ネットワーク環境を提供する．無線エミュレーション環境コンポーネントは，GPS 環境コンポーネントと同様に，当該時刻の移動体の位置情報から計算された通信遅延や帯域をネットワークデバイスドライバに設定しておくことによって，アプリケーション透過にエミュレーション機能を実現する．通信遅延や帯域の推定には，qomet 無線エミュレーションプログラムを利用している．

5. 評価

本章では，我々の提案する大規模アドホックネットワークシミュレーション環境のスケーラビリティを評価する．

評価は 2 種類ある．1 つは我々の提案する仮想マシンの配置手法のスケーラビリティに関する評価である．もう 1 つは，シミュレーション実行にともなう仮想マシンの生成時間や移動時間に関する評価である．

スケーラビリティ評価ではクラウド環境における既存のスケジューリング手法を使った場合と，我々の提案するボロノイ分割を使った仮想マシンの配置手法を比較する．3 章で説明したとおり，仮想マシンを用いたアドホックシミュレーションの問題点は，物理マシン間の通信輻輳である．よって，既存手法と比較して，物理マシン間をまたいだネットワーク通信がどの程度削減できるかを評価する．この評価によって，仮想マシンの配置に関する制約が解消されるので，大規模なアドホックネットワークシミュレーション環境の構築が可能であることを示すことができる．

我々の提案する仮想マシンの配置方法が，手法そのものの性質として，物理ネットワーク上に生成されるセッション数を削減し，仮想マシンによる大規模なアドホックネットワークシミュレーションを実現できるということを立証するためには，OpenStack などのデプロイツールに依存しない形で提案手法の評価が必要と考えた．したがって，次節に示すスケーラビリティ評価は，実 VM を起動する代わりに python のインスタンスを生成し，その中でアドホックルーティングの OLSRd に相当するプログラムを実行して，アドホックネットワークを形成している．

仮想マシンの生成時間や移動時間に関する評価では，まず，本提案手法のスケーラビリティ評価で用いたシナリオ上で発生した仮想マシンのライブマイグレーションの回数や頻度について評価する．次に，OpenStack による仮想マシンの生成やライブマイグレーション実行時間も評価する．これらの結果から，仮想マシンの生成や移動がシミュレーションの実行そのものには大きな影響を及ぼさないことを示す．

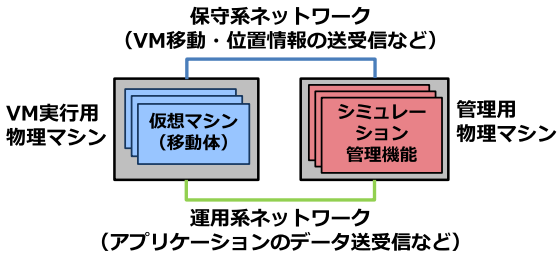

図 8 シミュレーション環境の構成

Fig. 8 Physical network configuration for simulation environment.

表 1 各物理マシンの仕様

Table 1 Specification of each physical machine.

CPU 数	8
メモリ量	32GB
OS	CentOS7.3
クラウドツール	OpenStack(PackStack)
ネットワーク	保守・運用系共に 1Gbps Ethernet

5.1 評価環境

図 8 にシミュレーション環境の構成，**表 1** に各物理マシンの仕様を示す．シミュレーション環境で利用する物理マシンは，管理用物理マシンと仮想マシン実行用物理マシンに分かれており，管理用物理マシンでは，OpenStack の管理機能のほかに，我々が作ったシミュレーション管理機能などが実行される．また，シミュレーション環境は 2 系統のネットワークを持つ．1 つは仮想マシンの移動や位置情報，ネットワーク遅延情報などの送受信を行う保守系ネットワークであり，もう 1 つはシミュレーションそのもので利用されるデータが送受信される運用系ネットワークである．

5.2 シミュレーションの実行手順

シミュレーションの実行手順としては，大きく次の 5 つの工程から構成される．

1. シミュレーションシナリオの作成
2. 移動体用 VM イメージの作成
3. マップ・移動データの作成
4. シミュレーション環境の実行
5. シミュレーションの開始

```
time:
      start : 0.0
      end : 717.8
map:
      file: map.net.xml
instances:
      - name: GPS
        base : gps
        coord :
          x : 34.21221
          y : 140.22151
        start : 0.0
        end : 717.8
      - name : vehicle_0
        route : trafficnet/dump_route/0.route
        start : 0.0
        end : 271.3
        device : gps
```

図 9 シミュレーションシナリオ例
Fig. 9 An example for simulation scenario.

表 2 移動体用 VM イメージの概要
Table 2 Virtual machine specification which is used as mobility.

CPU 数	1
メモリ量	128MB
OS	TinyCoreLinux
GPS モジュール等	Python スクリプト

シミュレーション環境の実行とシミュレーションの開始は自明であるため，次項以降で 1 から 3 工程の詳細を説明する．

5.2.1 シミュレーションシナリオの作成

シミュレーションシナリオは，シミュレーションで実行される移動体の生成時刻と終了時刻を記載したシナリオファイルである．**図 9** にシミュレーションシナリオの作成例を示す．シミュレーションファイルは大きく 3 つの部分から構成される．1 つ目（図中の time）はシミュレーションの開始時刻と終了時刻を記載した部分，2 つ目（図中の map）はシミュレーションで利用するマップファイルを記載した部分，3 つ目（図中の instances）は，シミュレーションで登場させる各移動体や固定物を具現化する仮想マシンの情報である．図に示すシナリオに従って，ある時刻になったら，起動に利用する仮想マシンのディスクイメージ（図中，base）を選択し，起動する位置情報（図中，coord，もしくは route）から起動に利用する物理マシンを選択して起動する．その後は仮想マシンがシミュレーション実行途中で集合区画間を移動する場合は，適宜ライブマイグレーションを実行することになる．

5.2.2 移動体用 VM イメージの作成

移動体用 VM イメージの作成とは，移動体を具現化するための仮想マシンに利用されるディスクイメージを作成することである．**表 2** に今回用意した移動体用 VM イメージの概要を示す．

TinyCoreLinux [17] は，Linux ディストリビューションの 1 つであり，OS サイズが 12 MB 程度と非常に小さいが，tce-load と呼ばれるパッケージ管理システムを備えており，他の組み込み系 Linux と比較しても汎用性が非常に高い．我々の提案手法では，仮想マシンの起動・終了・移動を頻繁に繰り返すため，OS は軽量で，かつ汎用性が高い TinyCoreLinux を選んだ．

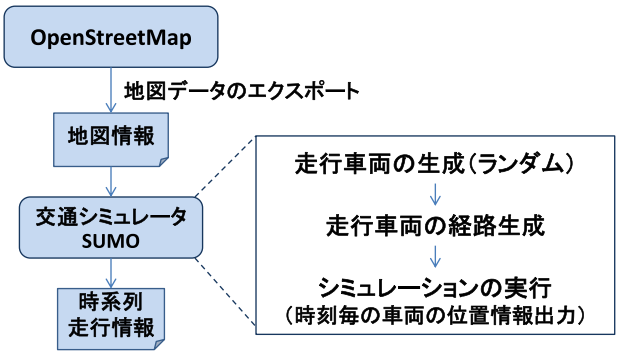

図 10 移動データの作成
Fig. 10 A sequence to make simulation parameters.

```
time, veh_id, vertical, horizontal
0.0, 0, 2292.85003801, 2518.65957292
0.1, 0, 2292.85446794, 2518.6400697
0.2, 0, 2292.86554278, 2518.59131164
...
```

図 11 時系列走行情報
Fig. 11 Time sequenced location data for each vehicle.

5.2.3 マップ・移動データ（位置情報）の作成

マップ・移動データの作成工程では，あらかじめ SUMO 交通シミュレータ [19] を使って，車両の時刻ごとの位置情報を生成する．また，マップデータは，自分で作るのではなく，フリーで公開されている道路情報である OpenStreetMap [18] からサンフランシスコ周辺の 2 × 2 km の区画の道路情報を SUMO へ入力した．**図 10** に交通シミュレータを使った車両位置情報の生成手順を示す．SUMO への入力は大まかに，1．道路情報，2．車両情報の 2 つから構成されている．車両情報は，SUMO に付属している車両の走行情報生成機能を使って車両の出現位置，走行ルートなどをランダムに生成している．

図 11 に実際に生成した時系列走行情報の一部を示す．時系列情報は，時刻（time），車両 ID（veh_id），経度情報（vertical），緯度情報（horizontal）から構成されている．経度と緯度情報は，実際のものとは異なり，シミュレータで使われている座標系となっている．この時系列走行情報は，図 9 のシミュレーションシナリオ中の route タグに紐づけられたファイル（trafficnet/dump_route/0.route）と

表 3 本項評価で使用したパラメータ
Table 3 Parameters which is used in this evaluation.

登場仮想マシン台数	100
区画分割の閾値	10
シミュレーション実行時間（秒）	700

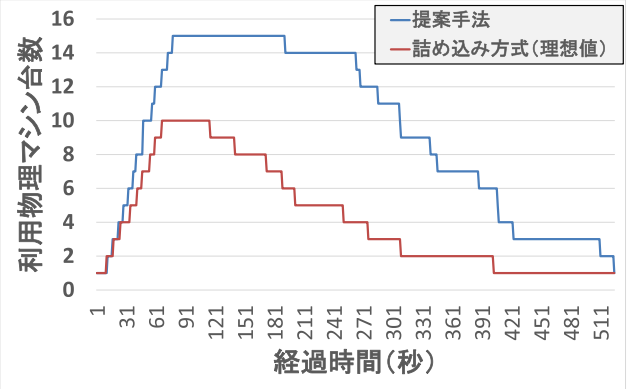

図 12 物理マシン利用台数の推移
Fig. 12 Time series transition graph how many physical machines are used in this simulation environment.

表 4 本項評価で使用したパラメータ
Table 4 Parameters which is used in this evaluation.

登場仮想マシン台数	1000
区画分割の閾値	10
シミュレーション実行時間（秒）	2000

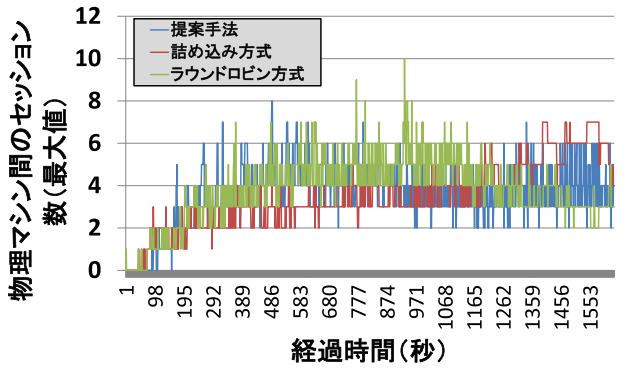

図 13 物理マシン間の最大セッション数の推移
Fig. 13 Time series transition graph of a number of maximum session in a physical network between two physical machines.

して利用される．

5.3 提案手法のスケーラビリティ評価

5.3.1 物理マシン利用台数に関する評価

まず，本項評価で使用したパラメータを**表 3** に示す．

図 12 は，クラウドの負荷分散手法として，提案方式と詰め込み方式を用いた場合の比較を表したグラフである．詰め込み方式とは，仮想マシンを物理マシンに詰め込んでゆき，仮想マシンの台数が，ある数になったら次の物理マシンを利用しはじめるという方法を指している．詰め込み方式は，物理マシンのリソースを無駄なく利用するという点においては理想的な方式といえる．一方，提案方式は，物理マシンで実行される仮想マシンの台数が上限に達していない場合でも，分割された区画の数だけ物理マシンを消費することになるので，仮想マシンを詰め込んでいった場合と比較すると，より多くの物理マシンを消費することになる．提案方式も詰め込み方式も，1 台の物理マシンに搭載できる仮想マシン数の上限を 10 台として，それを超えたら，提案方式は，区画を分割し，詰め込み方式は，次の物理マシン上で仮想マシンを起動する．

図を見ると，提案方式が詰め込み方式と比較して，より多くの物理マシンを消費していることが分かる．提案方式と詰め込み方式の利用物理マシン数の平均値は，それぞれ 9.28 台と 4.23 台であった．提案方式は，詰め込み方式と比較して，2 倍以上の物理マシンを利用していることになる．

今回，提案方式で用いた区画合併は，当該集合区画からすべての移動体がいなくなった場合にのみ区画を合併するという方法を用いている．よって，今回の評価結果は，提案手法が利用する物理マシン台数の最悪値といってもよい．

5.3.2 物理マシン間の最大セッション数に関する評価

まず，本項評価で使用したパラメータを**表 4** に示す．

物理マシン間の最大セッション数に関する評価は，仮想マシンを用いたシミュレーションにボトルネックとなる点が存在するのかどうかを確かめることが目的である．物理マシン間の最大セッション数とは，物理マシンどうしを接続する物理ネットワーク（ケーブル）のうち，最も多くセッションが作られるネットワークのことを指しており，ある物理ネットワーク上に作られるセッション数が極端に多くなると，当該ネットワークのデータ送受信が滞ってしまい，局所的なネットワーク輻輳が起こる．この結果としてシミュレーション全体の処理遅延を招くことになる．

図 13 は，提案手法と詰め込み方式，ラウンドロビン方式の物理マシン間の最大セッション数の推移を示している．提案方式と詰め込み方式，ラウンドロビン方式の最大セッション数の平均値は，3.517，3.626，3.753 である．このことからも，提案方式は他の方式と比較して，最大セッション数の平均値としてはそれほど差がないように見える．この結果からも，提案方式を用いても，ボトルネックを発生させるような物理マシン間の突発的なセッション数の増加は発生しないと考えている．

今回の評価において，提案手法は物理マシン 1 台を初期物理マシンとして，移動体が増えるたびに区画を分割して物理マシンを追加していった．提案手法では，集合区画を分割する際に，それまで同区画で通信していた移動体が 2 つに分割されるため，物理マシン間で生成されるセッション数が増加する．初期物理マシン数を数台程度に増やして

表 5 本項評価で使用したパラメータ
Table 5 Parameters which is used in this evaluation.

登場仮想マシン台数	1000
区画分割の閾値	10
シミュレーション実行時間（秒）	2000

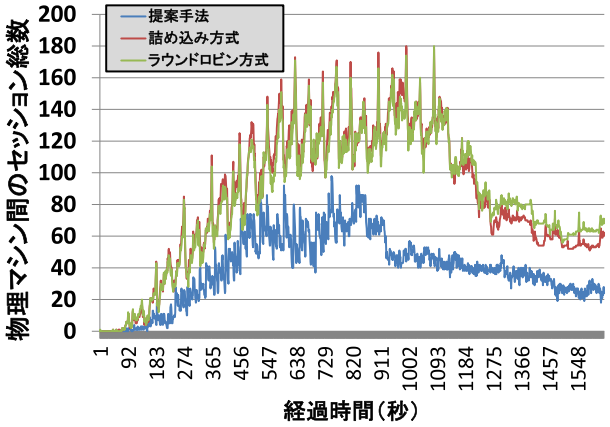

図 14 物理マシン間のセッション総数の推移
Fig. 14 Time series transition graph of total sessions in all physical network per each second.

表 6 本項評価で使用したパラメータ
Table 6 Parameters which is used in this evaluation.

登場仮想マシン台数	1150
区画分割の閾値	10
シミュレーション実行時間（秒）	2000

区画の分割を減らすと，提案方式は，セッション数の上昇をさらに軽減することが可能になると考えている．

5.3.3 物理マシン間のセッション総数に関する評価

まず，本項評価で使用したパラメータを**表 5** に示す．

物理マシン間のセッション数の総数に関する評価は，データセンタ全体のネットワーク輻輳をどの程度軽減することができるかということを確かめることが目的である．**図 14** は，提案手法と詰め込み方式，ラウンドロビン方式の物理マシン間のセッション総数の推移を示している．提案方式と詰め込み方式，ラウンドロビン方式のセッション総数の平均値は，それぞれ 32.28，78.88，77.17 であった．提案方式と比較して，詰め込み方式，ラウンドロビン方式の双方ともが約 2.4 倍程度のセッション数を持つことになった．

5.3.4 物理マシン間のセッション数とシミュレーションに登場する仮想マシンの台数の関係

まず，本項評価で使用したパラメータを**表 6** に示す．

図 15 は，物理マシン間のセッション数とシミュレーションに登場する仮想マシンの台数の関係を表したグラフである．仮想マシンを増やしていくと，提案方式と既存方式の差は拡大していることが分かる．具体的には，300 台の仮想マシンが現れる状況では，その差が 2 倍程度であったのに対して，1,000 台の仮想マシンが現れる状況では，既存方式は約 3.5 倍のセッションを物理マシン間に形成することが分かった．これは，提案方式が，地理的に近傍の移動体を 1 つの物理マシン上で実行させることによって，アドホックネットワークによって形成するセッションの大部分を，うまく物理マシン内に閉じ込めていることが，結果として表れていると考えている．図中，235 台の仮想マシンが実行されている部分で傾きが変化している．これはアドホックネットワークを形成できる距離に存在する車両がそもそも少なかったことから，提案方式も既存方式も，生成したセッション数に顕著な差が生まれなかったことに起因している．

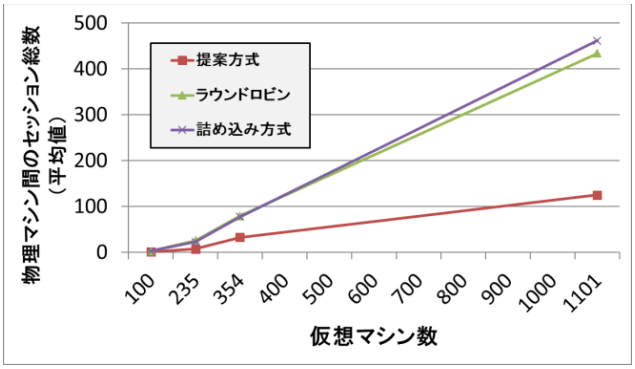

図 15 セッション数と仮想マシン台数の関係
Fig. 15 Relationship between a number of sessions and virtual machines.

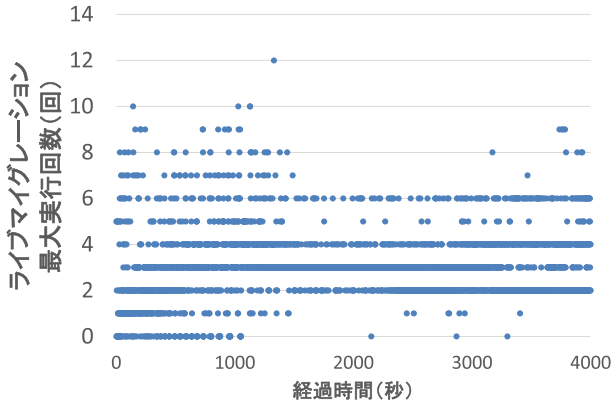

図 16 ライブマイグレーションの最大実行回数
Fig. 16 Time series transition graph of a number of live migrations executed in each seconds.

以上の結果から，提案方式が，突発的なセッション数の増加に関しては，シミュレーション実行時のネットワーク輻輳軽減に効果があることが明らかとなった．

5.4 仮想マシンの生成時間や移動時間に関する評価

本節では，仮想マシンの生成時間や移動時間を評価する．**図 16** は，2,000 台の移動体を実行するシミュレーションの中で発生したライブマイグレーション回数を 1 秒間隔で物理マシン別（厳密には 2 つの物理マシンの組合せごと）

表 7 OpenStack における仮想マシン生成時間
Table 7 Creation time of virtual machine executed by OpenStack.

	1 台	5 台	12 台
仮想マシン生成時間(秒)	1.83	5.22	8.96

表 8 OpenStack におけるライブマイグレーション時間
Table 8 Live migration time executed by OpenStack.

	1 台	5 台	12 台
仮想マシン移動時間(秒)	1.8	19.22	30.24

に集計し，そのうち，最も多かった回数を取り上げてグラフ化している．グラフを見ると，最大回数が 12 回，つまり，ある物理マシン間で 1 秒あたり 12 回のライブマイグレーションが実行されているということになる．一方，全物理マシンの平均は 1.8 回であることが分かった．

表 7 に OpenStack における仮想マシンの生成時間の評価結果を示す．仮想マシンの生成は 1 台あたり約 2 秒程度かかる．仮想マシンの生成は並行して実行されるため，台数が多くなると線形に時間が増加するということはなく，仮想マシンを 12 台生成してもかかる時間は約 9 秒程度で収まる．

表 8 に，OpenStack におけるライブマイグレーションの実行時間評価結果を示す．仮想マシンの移動は，1 台あたり約 2 秒程度かかる．一方，同時に移動する仮想マシンの台数が増えると，指数関数的に移動完了時間が増加する．仮想マシンの移動で問題になるのは，移動中の当該仮想マシンの停止状態である．仮想マシンが停止すると，当該仮想マシンを経路として利用しているネットワークに遅延時間が発生してしまう．文献による OpenStack のライブマイグレーションの評価 [20] によると，ライブマイグレーション中の仮想マシンの停止時間は，移動する仮想マシンのメモリサイズに関係なく，約 0.2 秒程度である．

本節の結果をまとめると，次のようなことがいえる．2,000 台の仮想マシンが参加するシミュレーションでは，ある物理マシン間で，最大 12 台の仮想マシンがマイグレーション中となることがあり，それら 12 台のマシンのうちのどれかを経路としている通信は，OpenStack をデプロイツールとして利用していると，0.2 秒程度の遅延をともなう可能性がある．

6. 考察

6.1 スケーラビリティに関する評価について

本節では，これまで行ったシミュレーション環境の評価について考察する．まず，仮想マシンを利用したアドホックネットワークシミュレーション環境のスケーラビリティとは以下の 4 つに集約できると考えている．

1. 物理マシン台数を増やすのと比例して仮想マシンの実行台数も増やすことが可能であること．
2. 仮想マシンの台数が増えた場合でも局所的なネットワーク輻輳を発生させないこと．
3. 仮想マシンの台数が増えた場合でもクラウド全体のネットワーク輻輳を発生させないこと．
4. 仮想マシンの生成や移動によってシミュレーション結果に影響を及ぼすことがないこと．

表 9 スケーラビリティ評価まとめ
Table 9 Summary of scalability evaluation.

	提案手法	既存手法
条件 1（台数拡張）	△	○
条件 2（局所輻輳低減）	○	△
条件 3（全体輻輳低減）	○	×
条件 4（生成・移動の影響）	△	×

これらの条件について，従来手法と我々が提案する手法の比較を**表 9** に示す．

条件 1 について，我々の提案手法は，物理マシンを増やすと，必ず CPU やメモリなどのリソースを使いきる形で仮想マシンの台数を増やすことができるわけではないので，既存手法よりは劣る．この条件については改善の余地があることについて次章で説明する．条件 2 について，提案手法と既存手法に物理マシン間に張られるセッション数の平均としては大きな差はないが，既存方式は間欠的にセッション数の最大値が，我々の提案手法を上回ることがあった．このことから，局所的なネットワーク輻輳の低減効果に関しては提案手法が優れているといえる．条件 3 については明らかに，我々の提案手法の優位性を示すことができたと考えている．条件 4 について，既存手法は，どの仮想マシン間でセッションが形成されているかという情報を持っておらず，クラウド環境の動的な負荷分散にライブマイグレーションを利用すると，ネットワーク輻輳を悪化させてしまう可能性が高いので，そもそも仮想マシンの移動を負荷分散の手段として利用することができない．したがって，条件 4 については既存手法に × をつけた．

以上の結果により，提案手法は既存の手法と比較して，シミュレーション環境の高いスケーラビリティを持つと我々は考えている．

6.2 仮想マシンの生成時間や移動時間に関する評価について

本節では，仮想マシンの生成時間や移動時間に関する評価についてまとめる．5.4 節での結果をふまえると，これまでの議論は次の 4 つにまとめることができる．

1. 仮想マシンのライブマイグレーションなどの通信は，保守系ネットワークに閉じて実行されており，シミュレーションで利用されるネットワークとは物理的に切り離されている．

2. 評価の結果，最大 12 台の仮想マシンが同時にマイグレーションするような状況が発生しうるが，12 台の仮想マシンが同時に移動するケースは，シミュレーション総時間のうち，0.025%であり，平均だと 1 秒間に 1〜2 台程度である．
3. OpenStack で実装されているライブマイグレーションでは，ライブマイグレーション自体の完了時間は指数関数的に増加するが，移動中の仮想マシンの停止時間は約 0.2 秒ときわめて小さい．
4. 文献 [21] によると，日本の交通密度は 1 平方キロメートルあたり約 150 台程度である．実験で用いた 2,000 台という車両台数は，非常に高密度な状況であり，実際のアドホックネットワークで利用される車両台数は，より少なくなる可能性が高い．

以上 4 つの状況から，区画移動中に発生するライブマイグレーション処理がシミュレーション結果に及ぼす影響は限定的であると考えている．

7. 今後の課題

7.1 実際にどこまで仮想マシン台数を増やすことができるのか？（実アプリケーションでの実証の必要性）

我々の提案方式は，1,000 台の仮想マシンを実行する場合，物理マシン間で生成されるセッションの総数を従来方式と比較して 3 分の 1 以下に抑えることが可能であることを示した．セッション生成数を低減させたことによって，これまでより大規模に仮想マシン上でアドホックネットワークを生成することが可能になったが，実際にどの程度まで仮想マシンを増やしていくことができるかは，想定するユースケースによって異なる．たとえば，災害を想定した緊急連絡網形成であれば，電話，もしくは SNS などの実アプリケーションを用意して，それらを使って評価する必要がある．

7.2 区画分割・区画合併パラメータの最適化

今回，提案方式で用いた区画合併は，当該区画からすべての移動体がいなくなった場合にのみ区画を合併するという方法を用いている．当該区画にある程度移動体が残っている状態で区画合併を実施し，当該区画に残っている移動体は近隣の区画を担当する物理マシンへマイグレーションするというような手法を用いれば，仮想マシンのマイグレーションコストと引き換えに，利用する物理マシン数をより削減し，詰め込み方式が利用する物理マシン数に近づけることができると考えている．

7.3 他の分野への応用（自動運転制御など）

今回の提案方式では，移動体の経路計算に交通シミュレータの SUMO を利用している．我々の提案するシミュレーション環境は，移動体（ノード）内部で任意のアプリケーションを動作させることが可能なので，SUMO などで外部から経路情報を入力する代わりに，自律的に移動方向や場所を決定できる機械学習機能を搭載することも可能である．このような機械学習機能を搭載した移動体が，他の移動体（ノード）と通信しつつ，近隣の移動体との間の通信帯域を最大化するように，移動体自身の位置を自律的に変化させることによって，より効果的なアドホックネットワークを作ることも可能になると考えている．

8. まとめ

本稿では，アドホックネットワークが形成する経路情報の局所性を生かして，地理的に近い場所にいる仮想マシンを同じ物理マシンに配置することによって，物理ネットワーク上に形成される通信路を低減させ，大規模な仮想マシンによるアドホックネットワークシミュレーション環境が構築可能であることを示した．

また，実環境と同じ環境で仮想マシン内のアプリケーションを実行するために，シミュレーションフレームワークとして必要なコンポーネントとアーキテクチャを提示した．

我々の研究によって，数千台から数万台規模の仮想マシンが動作する大規模アドホックネットワークシミュレーションを実現し，広域の災害に対しても，実用的なアドホックネットワークが形成できるということを立証することが可能となり，今後の災害対策提言に役立つものと考えている．

参考文献

[1] StarBED4 プロジェクトウェブサイト，北陸 StarBED 技術センター（オンライン），入手先 ⟨http://starbed.nict.go.jp/⟩（参照 2017-12-20）．
[2] 石田 亨，寺野隆雄，鳥居大祐ほか：社会シミュレーションと参加型デザイン，*IPSJ Magazine*, Vol.48, No.3, pp.271–277 (2007).
[3] 中央防災会議：南海トラフ巨大地震の被害想定について，内閣府（オンライン），入手先 ⟨http://www.bousai.go.jp/jishin/nankai/taisaku_wg/pdf/20130318_shiryo2_1.pdf⟩（参照 2017-12-20）．
[4] Asterisk, Asterisk ウェブサイト（オンライン），入手先 ⟨https://www.asterisk.org/⟩（参照 2018-04-13）．
[5] NS3, NS3 ウェブサイト（オンライン），入手先 ⟨https://www.nsnam.org⟩（参照 2017-12-20）．
[6] Varga, A.: OMNet++ Discrete Event Simulation System, OMNET++ ウェブサイト（オンライン），入手先 ⟨http://omnetpp.org/⟩（参照 2017-12-20）．
[7] Cloonix project, Cloonix project ウェブサイト（オンライン），入手先 ⟨http://www.cloonix.net⟩（参照 2017-12-20）．
[8] Pizzonia, M. and Rimondini, M.: Netkit: Easy emulation of complex networks on inexpensive hardware, *Proc. 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities* (2008).
[9] CORE：Common Opensource Routing Emulator, CORE ウェブサイト（オンライン），入手先 ⟨https://www.

nrl.navy.mil/itd/ncs/products/core⟩（参照 2017-12-20）.
[10] Apostolopoulos, G. and Hassapis, C.: V-eM: A Cluster of Virtual Machines for Robust, Detailed, and High-Performance Network Emulation, *14th IEEE International Symposium on Modeling, Analysis and Simulation*, pp.117–126 (2006).
[11] Weing, E., Schmidt, F., Lehn, H., et al.: SliceTime: A platform for scalable and accurate network emulation, *8th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI 11*) (2011) (online), available from ⟨http://www.usenix.org/events/nsdi11/tech/slides/weingartner.pdf⟩.
[12] Grau, A., Maier, S., Herrmann, K. and Rothermel, K.: Time jails: A hybrid approach to scalable network emulation, *Proc. Workshop on Principles of Advanced and Distributed Simulation* (*PADS*), pp.7–14 (2008).
[13] Bergstrom, C., Varadarajan, S. and Back, G.: The distributed open network emulator: Using relativistic time for distributed scalable simulation, *Proc. Workshop on Principles of Advanced and Distributed Simulation*, pp.19–25 (2006).
[14] Yoginath, B., Perumalla, S. and Henz, J.: Virtual machine-based simulation platform for mobile ad-hoc network-based cyber infrastructure, *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol.12, No.4, pp.439–456 (2015).
[15] Beuran, R., Nakata, J., Okada, T., Nguyen, L.T., Tan, Y. and Shinoda, Y.: A multi-purpose wireless network emulator: QOMET, *Proc. International Conference on Advanced Information Networking and Applications*, pp.223–228 (2008).
[16] Zhao, Y. and Huang, W.: Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud, *5th International Joint Conference on INC, IMS and IDC*, pp.170–175 (2009).
[17] TinyCoreLinux, TinyCoreLinux ウェブサイト（オンライン），入手先 ⟨http://distro.ibiblio.org/tinycorelinux/⟩（参照 2018-04-25）.
[18] OpenStreetMap, OpenStreetMap ウェブサイト（オンライン），入手先 ⟨https://www.openstreetmap.org⟩（参照 2017-12-20）.
[19] SUMO – Simulation of Urban Mobility, Institute of Transportation Systems (online), available from ⟨http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/⟩ (accessed 2017-12-20).
[20] Performance analysis of “post-copy” live migration in OpenStack, Zurich University of Applied Sciences (online), available from ⟨https://blog.zhaw.ch/icclab/performance-analysis-of-post-copy-live-migration-in-openstack/⟩ (accessed 2018-04-29).
[21] 太田貴志，ゲルハルト・ロルマン：24 Ghz 帯 UWB 近距離レーダ搭載車両の日本事情における普及予測の初期検討，総務省（オンライン），入手先 ⟨http://www.soumu.go.jp/main_sosiki/joho_tsusin/policyreports/joho_tsusin/uwb_wlsystem/pdf/070523_1_sa3.pdf⟩（参照 2018-04-29）.

小比賀 亮仁

北陸先端科学技術大学院大学．日本電気株式会社．

篠田 陽一

北陸先端科学技術大学院大学．

岩井 孝法 （正会員）

日本電気株式会社．

里田 浩三

日本電気株式会社．