2007－OS－106 (16)
2007／8／3

通信の揺らぎを考慮したオーバーレイ開発支援環境

島田明男　浅原 理人　河野 健二

慶應義塾大学 理工学部 情報工学科
E-mail: {reds, asahara}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

大規模な分散システムではサービスの可用性や耐故障性を高めるために，オーバーレイネットワークという技術を利用している．オーバーレイネットワークとは分散アプリケーションによって構築される仮想的なネットワークである．オーバーレイネットワークが検索やマルチキャストといった機能を提供することで，効率的なサービスの運用が可能になる．このようなオーバーレイネットワークを開発・評価するための環境が開発者には必要となるが，実際に大規模なネットワークを用意し，実験を行うのはコスト面などを考慮すると困難である．本論文ではオーバーレイネットワークに特化した開発支援環境として NetCamp を提案する．NetCamp では物理ネットワーク上で発生する輻輳やパケットロスの影響をエンドホスト間の通信遅延に集約し，スケーラビリティを確保する．また，通信遅延を擬似的に再現する機構をモジュール化することで，開発環境の使用者が通信遅延の揺らぎを自由に設定できる．NetCamp の有用性を確認するため，分散システム上のノード間の Round-Trip-Time を予測する技術である Network Coordinate System を NetCamp 上に実装し，評価を行った．実験の結果，Network Coordinate System の特徴を NetCamp によって評価することができた．また，その際約 1,000 台のノードの挙動を高い精度を保ちながら再現することができた．

An Environment for Developing Overlay Networks with latency Fluctuation

Akio Shimada　Masato Asahara　Kenji Kono

Department of Information and Computer Science, Keio University
E-mail: {reds, asahara}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

For large-scale distributed applications, constructing an overlay network is a promising approach to enhancing their availability and fault-tolerance. The overlay network is a virtual network that a distributed application builds independently from the underlying network. Various functionalities such as information retrieval and multicast have been constructed with the overlay to improve efficiency of service management. However, it is quite difficult to develop and evaluate the overlay network because preparing a large-scale network is a costly and often unrealistic task. This paper proposes NetCamp, a support environment for developing and evaluating the overlay network. NetCamp aggregates various network events such as network congestion and packet loss into the latency of end-to-end hosts and simplifies the process of emulating those network events. This simplicity enables NetCamp to emulate a large number of nodes on a single physical machine. With NetCamp, we developed and evaluated Network Coordinate Systems which predict Round-Trip-Time between nodes on distributed systems. We could observe the features of the Network Coordinate Systems. Moreover, NetCamp successfully emulated the behavior of about 1,000 nodes.

1 はじめに

Peer-to-Peer 技術を用いた大規模な分散システムによるネットワークサービスの提供が盛んになっている [1, 2, 3, 4, 5, 6]．大規模な分散システムではサービスの可用性，耐故障性を高めるためにオーバーレイネットワークという技術を利用している．オーバーレイネットワークとは分散アプリケーションによって構築される仮想的なネットワークである．物理ネットワークのトポロジから独立した独自のトポロジをアプリケーションレイヤで構築する．オーバーレイネットワークが検索やマルチキャストといった機能を提供することで，効率的なサービスの運用が可能になる．現在では提供するサービスの質をさらに向上するため，エンドホスト間の通信遅延を考慮したオーバーレイネットワーク構築手法も提案されている．例えば Network Coordinate System[7, 8] は分散システムを構築するノードのエンドホスト間の通信遅延を推定する機能を提供する．この機能を用いて，各ノードが通信遅延の小さいノードを選択することにより，ファイルのダウンロード時間の短縮などを図ることができる．また，通信遅延を考慮したルーティングを行う分散ハッシュテーブル (DHT)[5, 9] なども存在する．

オーバーレイネットワークを開発するうえで，開発者は実装，評価，デバッグといった手順を踏むことになる．しかし，実際にサーバを各所に分散配置し，実験を行うのは困難である．オーバーレイネットワークは数千から数万のノードで構築されること

も多い．例えば，オーバーレイネットワークを用いたコンテンツ配信基盤であるAkamai[2]は約14,000台のサーバを実際に分散配置し，サービスを運用している．このような大規模なネットワークを用意することは不可能ではないがそのコストはとても大きい．そこでオーバーレイネットワークの開発・評価を支援することが可能なスタンドアロンな環境が求められている．

既存の開発環境には大きくわけて，1) 物理ネットワークをシミュレート，エミュレートするもの [10, 11]，2) オーバーレイ構築のフレームワークを提供するもの [12]，3) 実機を広域分散配置して実際に動作させるものがある [13]．1) では物理ネットワークで起きる輻輳やパケットロス等を再現することができる．その反面計算機の負荷が大きくなり，挙動を再現可能なノード数が制限される．オーバーレイネットワークの評価に用いるにはスケーラビリティが低い．2) のような開発環境のなかには，エミュレーション機能を用いることで，繰り返し動作検証を行いながら大規模なオーバーレイネットワークの開発を行えるものが存在する．しかし，動作検証の際にノード間の通信遅延はエミュレーションされない．従来のオーバーレイネットワークは通信遅延を考慮していなかったため，オーバーレイ上のルーティングのホップ数やメッセージ数でのみ評価を行えばよかった．しかし，前述したとおり，近年は通信遅延を考慮したオーバーレイネットワークが多々提案されている．このようなオーバーレイネットワークはノード間の通信遅延を再現しなければ正確な評価を行うことはできない．3) では実際のインターネット上でオーバーレイネットワークの評価を行うことが可能である．しかし，実際のインターネットを用いているため，ネットワークのバーストやリンクの切断などのような状況を意図的に発生させオーバーレイネットワークを評価することができない．

本論文ではオーバーレイネットワークに特化した開発支援環境としてNetCampを提案する．NetCampではオーバーレイネットワークを評価する際に重要となる以下の3点を実現する．

- スケーラビリティ：より多くのノードの挙動を再現可能である
- 通信遅延の再現：エンドホスト間の通信遅延を擬似的に再現可能である
- ネットワークの可変性：開発環境上で再現する擬似的なネットワークの状況を意図的に変化させ，様々な状況下でオーバーレイネットワークの評価ができる

NetCampでは通信コストを考慮したオーバーレイネットワークの評価を可能にするため，ノード間の通信遅延を擬似的に再現する．その際，物理ネットワーク上で起きる輻輳やパケットロスといった現象の影響をエンドホスト間の通信遅延に集約することで，計算量を下げ，スケーラビリティを確保する．NetCampでは通信遅延の揺らぎをあたえる機構をモジュール化し，NetCampの利用者が遅延の揺らぎを自由に設定することを可能にする．これによって通信遅延を擬似的に再現し，ネットワークの可変性を確保する．作成したモジュール次第で，様々なネットワークの状況を意図的に発生させ，オーバーレイネットワークの評価を行うことができる．

NetCampの有効性を示すため，Network Coordinate SystemであるGlobal Network Positioning[7]とVivaldi[8]をNetCamp上に実装し比較を行った．比較の結果，各Network Coordinate Systemを提案した文献が示す特徴をNetCamp上で評価することができた．また，その際約1,000台のノードの挙動を高い精度を保ちながら再現することができた．

以下，第2章ではオーバーレイネットワークのための開発・評価環境に求められる要件を述べる．第3章ではNetCampの概要について述べる．第4章ではNetCampがオーバーレイネットワークの評価に有用である一例を示す．第5章では関連研究について述べる．最後に第6章で本論文をまとめる．

2 設計要件

本章ではオーバーレイネットワークについて述べた後，オーバーレイネットワークの開発支援環境に求められる要件を明確にする．

2.1 オーバーレイネットワーク

従来のオーバーレイネットワークではオーバーレイネットワーク上の距離と物理ネットワークでの経路上の距離は完全に独立したものとして扱われていた．よって，オーバーレイ上では最適なルーティングでも，物理ネットワーク上では遠回りになる状況が発生した．このような問題に対処するため，エンドホスト間の通信遅延を考慮し，より効率的なオーバーレイネットワークを構築する手法が提案されている．

コンテンツ配信基盤であるCoralCDN[5]ではコンテンツのキャッシュ検索にDHT[14]を利用している．DHT上でルーティングを行う際に，各ノードは

コンテンツ検索のリクエストの転送先として，RTTの小さいノードを優先的に選択する．こうすることで，オーバーレイ上でのルーティングにエンドホスト間の通信遅延を反映し，検索時間を短縮できる．また，同じくコンテンツ配信基盤であるAkamaiではクライアントからのリクエストをクライアントからの通信遅延が小さいサーバにリダイレクトすることで，クライアントがサービスを受けるまでの時間の短縮を図っている．

2.2 開発支援環境の要件

オーバーレイネットワークを評価するために，開発支援環境はいくつかの要件を満たす必要がある．ここではその要件について述べる．

2.2.1 スケーラビリティ

オーバーレイネットワークのための開発支援環境にはスケーラビリティが求められる．オーバーレイネットワークは大規模な分散システム上に構築することを想定されている．オーバーレイネットワークによるコンテンツ検索技術やマルチキャスト技術は大規模な分散システム上でサービスの可用性や耐故障性を高めるために開発されてきた．よって，オーバーレイネットワークの評価を正確に行うためには，より多くのノードの挙動を開発支援環境上で再現する必要がある．

2.2.2 通信遅延の考慮

従来のオーバーレイネットワークではオーバーレイネットワーク上の距離と物理ネットワークでの経路上の距離は完全に独立したものとして扱われていた．よって，開発支援環境上でオーバーレイネットワークの評価を行う際に通信遅延を考慮する必要はなく，ルーティングのホップ数やメッセージ数を評価の基準とするだけで十分とされていた．しかし，現在ではエンドホスト間の通信遅延を考慮したより効率的なオーバーレイネットワーク構築手法が提案されている．エンドホスト間の通信遅延を物理的ネットワークの経路上の距離とみなし，オーバーレイネットワーク上でのルーティングやマルチキャストに反映することでより効率の良いサービスを提供することができる．このようなオーバーレイネットワークではルーティングのホップ数やメッセージ数による評価だけでは不十分である．開発支援環境上でエンドホスト間の通信遅延を再現しなければ，通信遅延を考慮したオーバーレイネットワークを正しく評価することができない．

2.2.3 ネットワークの可変性

エンドホスト間の通信遅延を考慮したオーバーレイネットワークは通信遅延の揺らぎによる影響を受ける．この通信遅延の揺らぎは物理ネットワークの状況の変化によって引き起こされる．ネットワークの輻輳や障害によるリンクの切断などの影響を受けエンドホスト間の通信遅延は大きく変化する．エンドホスト間の通信遅延の揺らぎが大きくなると，オーバーレイネットワーク上の距離に物理ネットワークの経路上の距離を反映させるのが困難になる．よって，オーバーレイネットワークの評価，デバッグを行うためには様々なネットワークの状況を想定して，オーバーレイネットワークのテストを行う必要がある．開発支援環境上で擬似的なネットワークの状況変化を意図的に発生させ，エンドホスト間の通信遅延の揺らぎに対して，オーバーレイネットワークの挙動がどう変化するか評価する機能が開発支援環境に求められる．

3 NetCamp

本章ではNetCampの概要について述べる．前章で明確にしたオーバーレイネットワーク評価のための要件である，スケーラビリティの確保，通信遅延の再現，ネットワークの可変性をどのように実現するかを説明する．

3.1 設計

3.1.1 スケーラビリティの確保

NetCampでは物理ネットワーク上で起きる輻輳やパケットロスといった現象の影響をエンドホスト間の通信遅延に集約することで，開発支援環境のスケーラビリティを確保する．図1はオーバーレイネットワークと物理ネットワークの違いを示す．ネットワークシミュレータやネットワークエミュレータのような従来の開発支援環境は図1の下部で示すようなオーバーレイネットワークの下位の物理ネットワークまでシミュレート，エミュレートすることで，物理ネットワークで起こる輻輳やパケットロスといった現象の再現が可能になる．しかし，物理ネットワークのシミュレーション，エミュレーションのために，計算機の負荷が高くなり，スケーラビリティが低下してしまう．これに対し，NetCampではエンドホスト間の通信遅延のみを擬似的に再現する．物理ネットワークのシミュレーション，エミュレーションを行わない分，計算機の負荷が軽減され，従来の開発支援環境よりも高いスケーラビリティを確保することができる．

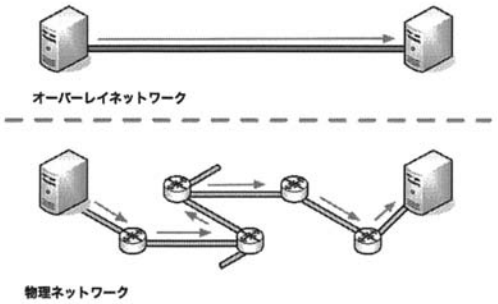

図 1: オーバーレイネットワークと物理ネットワーク

この最適化はオーバーレイネットワークの評価において妥当である．オーバーレイネットワークはアプリケーションレイヤに構築される仮想的なネットワークである．オーバーレイネットワークを構築する分散アプリケーションはエンドホスト上で実行されている．よって，オーバーレイネットワークが扱うのはノード間の RTT のような，エンドホスト上で計測可能な情報のみである．物理ネットワーク上の輻輳やパケットロスといった情報をオーバーレイネットワークが扱うことはない．したがって，オーバーレイネットワークを開発支援環境上で構築するためには，物理ネットワークで起きている現象まで再現する必要はなく，エンドホスト間の通信遅延のみを再現すればよいと考えられる．

3.1.2 通信遅延の再現

NetCamp ではノード間でメッセージの送受信をする際に，エンドホスト間の通信遅延を擬似的に再現する．通信遅延を再現する機構のモデルを図 2 に示す．送信側と受信側のノードは一対ごとに通信を行うためにバッファを共有している．送信側のノードがデータを共有バッファに書き込み，それを受信側のノードが読み込むことで，通信が行われる．送信側はまず，書き込むデータを遅延装置に渡す．遅延装置は指定された遅延の後，データを共有バッファに書き込む．このようにして，NetCamp ではエンドホスト間の通信遅延を再現する．

3.1.3 ネットワークの可変性

NetCamp が擬似的に再現する通信遅延は NetCamp の使用者が自由に設定できる．通信遅延を与える機構をモジュール化し，そのモジュールを NetCamp の使用者が作成することで，これを実現する．この機能により，NetCamp の使用者はモジュールの作り方次第で，ネットワークのバーストやリンクの切断といった，ネットワークの状況の変化を意図的に発生させ，オーバーレイネットワークの評価を行うことができる．一度オーバーレイネットワークのアプリケーションモデルを NetCamp 上に実装してしまえば，モジュールを差し替えるだけで様々なネットワークの状況を想定した実験・評価が可能になる．

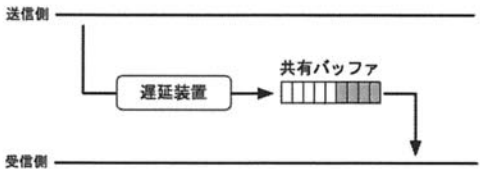

図 2: 遅延再現機構

例えば，ある分布に基づいて遅延の揺らぎを与えるモジュールを作り，その揺らぎを増幅させることで，オーバーレイネットワークが通信遅延の揺らぎにどこまで対応することができるかを検証することができる．また，インターネットから取得したログをもとにホスト間の遅延やスループットを変化させるモジュールを作り，実際のインターネットに近い環境でオーバーレイネットワークを動作させたときの評価を行うこともできる．

3.2 実装

NetCamp を Java 上のフレームワークとして実装した．アプリケーションモデルの構築を容易にするため，NetCamp を次のように実装した．

NetCamp は図 3 に示すコンポーネントによって構成されている．NetCamp では挙動を再現する仮想ノードをスレッドとして表現する．ノード間のメッセージの送受信をスレッド間のメッセージの送受信としてエミュレートする．各スレッドにはホスト名が割り振られ，このホスト名を指定することで各仮想ノードは他の仮想ノードと通信を行う．ソケットインタフェースがスレッド間の通信に関するインタフェースを提供する．このインタフェースは Java の通常のソケットインタフェースに近いものになっており，NetCamp の利用者は通常のネットワークプログラミングを行う感覚でアプリケーションモデルを容易に構築することができる．遅延モジュールと遅延エミュレータはエンドホスト間の通信遅延を擬似的に再現する．遅延モジュールは NetCamp の使用者が作成する．

各仮想ノードが通信を行う際に送信したデータはソケットインタフェースにより，出力ストリームに渡される．出力ストリームは受け取ったデータを遅延エミュレータに渡す．遅延エミュレータは遅延モジュールから与えるべき通信遅延を取得し，その

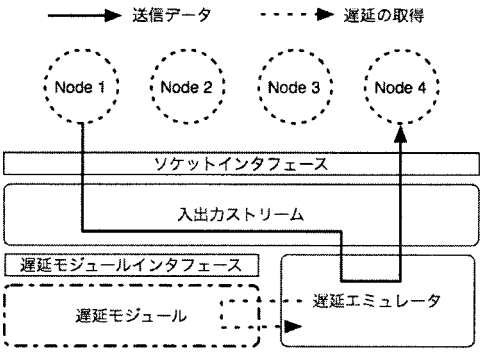

図 3: NetCamp の各種コンポーネント

遅延を擬似的に再現する．現在の実装ではタイマースレッドを用い，再現すべき遅延が経過した時刻に入出力ストリームの共有バッファにデータを書き込むことで通信遅延の再現を実現する．受信側の仮想ノードはソケットインタフェースを介し，入力ストリームからデータを受け取る．

3.2.1 遅延モジュール

遅延モジュールは getLatency，getRTT の 2 つの抽象メソッドを持つ．NetCamp の利用者はこの 2 つの抽象メソッドを実装することで，遅延モジュールを作成する．

getLatency メソッドは遅延エミュレータによって呼び出されるメソッドである．遅延エミュレータはこのメソッドにより，仮想ノード間の通信において与えられるべき遅延の値を取得し，その遅延を擬似的に再現する．getLatency メソッドは int 値を引数として取る．この値は送信するメッセージのサイズを表す．遅延エミュレータは getLatency を呼び出す際にメッセージサイズを出力ストリームから受け取り，getLatency の引数とする．遅延モジュールはこのサイズをもとにノード間の通信において与えられるべき遅延を計算し long 値として返す．遅延エミュレータはこの値を遅延として擬似的に再現する．遅延の精度はミリ秒単位である．

getRTT は仮想ノード間の RTT を計測するために用いられるメソッドである．仮想のノード間の RTT を計測するためのインタフェースとして，Ping クラスが用意されている．このクラスの exec メソッドは getRTT により，ノード間の RTT を取得する．その際 exec メソッドを呼び出したスレッドの実行を RTT の値だけミリ秒単位で停止する．

4 実験

NetCamp の有用性を示すための実験を行った．Network Coordinate System である GNP と Vivaldi を NetCamp 上に実装し，それぞれの特徴を評価した．また，その際に NetCamp 上で再現すべき遅延と実際に再現された遅延の誤差率を記録し，NetCamp 自体の精度を測定した．本章では各 Network Coordinate System の特徴について述べた後，それぞれの特徴を評価するために行った実験の結果を示す．Xeon 3.0GHz を 2 基，メモリを 16GB 搭載した計算機を用い実験を行った．OS には Free BSD 6.2，JVM は Java HotSpot(TM) 64-Bit Server VM を用いた．

4.1 Network Coordinate System

Network Coordinate System は分散システム上のノード間の距離を推定する手法のひとつである．各ノードにユークリッド距離がノード間の通信遅延の近似となるような M 次元座標を与える．これにより，ノード間の通信遅延を容易に求められる．

4.1.1 Global Network Positioning

Global Network Positioning(GNP) では，あらかじめランドマークとなるいくつかのノードを選択しておく．これらのランドマークに座標を与え，ランドマーク間の座標上の距離がランドマーク間の RTT の実測値に近くなるように座標を求める．ランドマーク以外のノードは各ランドマークまでの座標上の距離と，各ランドマークまでの RTT の実測値が近くなるように座標を求める．このようにして，GNP では各ノードに絶対的な座標を与える．

GNP ではシステムの初期化時に静的に各ノードの座標を決めるため，物理ネットワーク上での経路変化などにより，長期的に RTT が変動すると精度が低下する．

4.1.2 Vivaldi

Vivaldi は GNP のように，あらかじめランドマークを設定する必要がない，完全な Peer-to-Peer 方式の Network Coordinate System である．各ノードは隣接ノードと通信するたびに，ノード間の RTT を計測する．そして，自身と隣接ノードとの座標上の距離が RTT の実測値に近くなるように，自身の座標を修正する．これにより，ノード間の RTT の変動に追随しながら，ノード間のユークリッド距離を RTT に近似することができる．

Vivaldi は初期化時からしばらくは RTT の予測値が不正確だが，各ノードが隣接ノードと通信をするたびに座標の修正を行うので，徐々に精度は上昇する．また，物理ネットワークの状況変化により，長期的に RTT が変動しても高い精度を保つことができる．

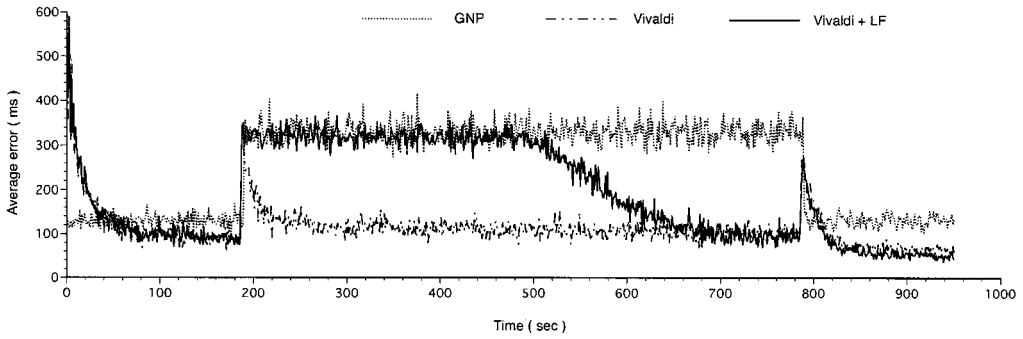

図 4: 長期的な RTT の変動によるエラー率の変化

4.1.3 Vivaldi + Latency Filter

Latency Filter(LF)[15] を用いることで Vivaldi の精度を高めることができる．Latency Filter はパルス的に発生する RTT の大きな変動を除去し，座標の精度を高める機能を持つ．Latency Filter はあらかじめ決めておいたウインドウサイズ分だけ，過去に計測した RTT を記録しておく．そして座標の修正を行う際，指定されたパーセンタイル値の RTT を RTT の実測値として採用する．例えばウインドウサイズが 8，パーセンタイルが 25%ならば，過去に記録しておいた RTT の中から，2 番目に小さい値を RTT の実測値として採用する．

Vivaldi ではパルス的に発生する RTT の大きな変動に追随し，自身の座標を修正してしまう．その結果 RTT の予測値の精度が低下してしまう．Latency Filter を用いれば RTT の異常値を検出し除去できるので，RTT が頻繁に変動するような状況でも高い精度を保つことができる．

4.2 RTT の長期的変動に対する特性評価

長期的な RTT の変動を NetCamp 上で発生させ，各 Network Coordinate System の性能を評価した．RTT のサンプルを取得するために，GT-ITM[16] を用いて Transit-Stub モデル [17] のトポロジを作成した．トポロジから 115 のエンドホストを抽出し，これらのホスト間の一覧表を 2 種類作成した．ひとつは通常のトポロジの一覧表，もうひとつはある Transit 間の RTT を 10 倍にしたトポロジの一覧表である．遅延モジュールは RTT を取得する 2 つの一覧表を実験の途中で差し替えることで，ネットワークの状況の長期的な変化を意図的に発生させた．

各ノードは 32 の隣接ノードを持ち，各隣接ノードと 1 秒の間隔で順番に通信を行う．生成する座標の次元数は 6 次元とした．実験開始から 200 秒後に通常のトポロジから Transit 間の RTT を 10 倍にしたトポロジに切り替え，800 秒後に元に戻した．各ノードが隣接ノードと通信するたびに，RTT の予測値と RTT の実測値の差の絶対値をエラーとして記録した．

実験結果を図 4 に示す．横軸は経過時間，縦軸は 1 秒ごとのエラーの平均値である．GNP はトポロジが変化した後，エラーの平均値が約 350ms にまで上昇したままである．これは，GNP は初期化時に静的に座標を決めてしまい，RTT の変動による座標の修正を動的に行わないためである．それに対し，Vivaldi はトポロジの変更直後はエラーの平均値が GNP と同程度上昇しているが，すぐに座標の修正を行い，約 50 秒後にはエラーの平均値が約 100ms に低下している．Vivaldi + LF はトポロジの変化後，しばらくエラーは上昇したままである．これはトポロジが変化してからしばらくは Latecy Filter が上昇した RTT を異常値として除去してしまうからである．

4.3 RTT の短期的変動に対する特性評価

通信遅延の揺らぎを変動させたときの，各 Network Coordinate System の性能を評価した．パレート分布に基づき RTT を変動させる遅延モジュールを作成し実験を行った．パレート分布はインターネット上のノード間の RTT の分布を良く近似することが知られている [18]. パレート分布は以下の式で与えられる．

$$F(x) = 1 - \left(\frac{k}{x}\right)^{\alpha}, x \geq k$$

α がパラメータになっており，この値が小さいほど，大きな値の RTT が出現する確率が高まり，揺らぎが大きくなる．

揺らぎを与えない場合と α の値を 5.0，3.0，1.0 と変えたときとで，それぞれ実験を行った．King

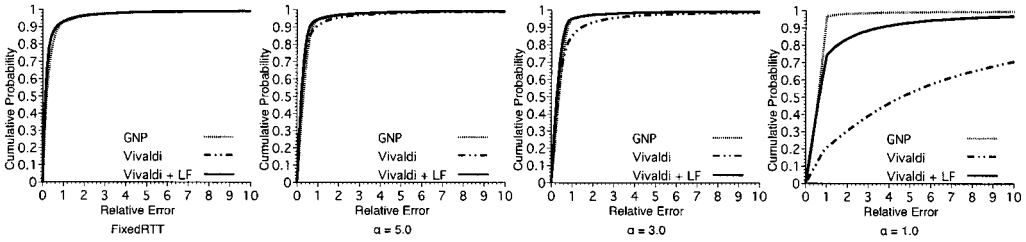

図 5: 短期的な RTT の変動によるエラー率の分布の変化

表 1: 4.2 節の実験における遅延の再現精度

GNP	0.6%
Vivaldi	0.6%
Vivaldi + LF	1.2%

表 2: 4.3 節の実験における遅延の再現精度

	揺らぎなし	α=5.0	α=3.0	α=1.0
GNP	2.9%	2.0%	2.6%	1.5%
Vivaldi	7.7%	8.7%	8.4%	5.6%
Vivaldi + LF	4.1%	7.5%	7.5%	3.7%

method[19] により得られた 1,024 台の DNS サーバ間の RTT の一覧表をもとに，パレート分布による揺らぎを遅延モジュールを用いて与えた．隣接ノード数，座標の更新間隔，座標系の次元数は 4.2 節と同様である．実験開始 500 秒後から 500 秒間，各ノードが隣接ノードと通信するたびに，RTT の予測値と RTT の実測値の相対誤差率を記録した．

実験結果を図 5 に示す．横軸は相対誤差率，縦軸は相対誤差率の累積確率である．RTT の揺らぎがないとき，α が 5.0 のときははどの Network Coordinate System も同程度の性能を示している．しかし，Vivaldi は α が 3.0 以下になると，他の 2 つに比べて，精度が低下しているのがわかる．Vivaldi は RTT の変動に応じて，座標を修正する機能をもっている．そのため，RTT の異常値にも敏感に反応してしまう．その結果，たとえ座標が最適であっても，RTT の異常値に追随し，座標を更新してしまい精度が落ちている．Vivaldi + LF は揺らぎが大きくなっても Vivaldi よりも高い精度を保っている．α が 3.0 ときでは相対誤差率 0.7 以内に 90%のアクセスが収まっている．α が 1.0 のときにも，Vivaldi に比べて高い精度を保っている．これは Latency Filter が RTT の異常値をフィルタリングしているため，座標の更新に RTT の異常値が反映されないからである．Latency Filter を用いれば，RTT の揺らぎが大きいときでも最適な座標を保つことができる．

4.4 遅延の再現精度

4.2 節と 4.3 節の実験において，NetCamp 上で再現すべき遅延と実際に再現された遅延の平均誤差率をそれぞれ，表 1 と表 2 に示す．4.2 の実験では平均して約 0.8%以内に，4.3 の実験では平均して約 5.2%以内に誤差率収まっている．高い精度で通信遅延を NetCamp 上で再現することができたといえる．

5 関連研究

ns-2[10] はネットワークアプリケーションの評価のために幅広く利用されているネットワークシミュレータである．物理ネットワークまでシミュレートしており，輻輳やパケットロスといった現象まで再現することが可能だが，その分計算量が大きくなり，シミュレーションで再現可能なノード数に限りがある．

PeerSim[20] は Peer-to-Peer ネットワークのプロトコル評価のためのネットワークシミュレータである．一台の計算機で数万台のノードの挙動をシミュレート可能である．しかし，ノード間の通信の際に通信遅延がシミュレートされないため，オーバーレイネットワークの評価の基準にルーティングのホップ数やメッセージ数を用いることしかできない．

p2psim[21] は PeerSim 同様，Peer-to-Peer ネットワークのプロトコル評価のためのネットワークシミュレータである．1 台の計算機で 3,000 台のノードのシミュレーションを現実的な実行時間で処理可能である．p2psim では通信を行うノード間の RTT の一覧表をもとに遅延をシミュレートする．シミュレートされる遅延時間は固定的なものであり，NetCamp のように，通信遅延に揺らぎを与えることはできない．

ネットワークエミュレータとして ModelNet[11] が挙げられる．ModelNet はネットワークをエミュレートする Core Node 群とネットワークアプリケー

ションを実行する Edge Node 群によって構成される．Edge Node 上のネットワークアプリケーションから送信されたパケットは Core Node 群でエミュレートされるネットワークを経て，他ノードに到達する．スケーラビリティは 1 台の Core Node と 10 台の Edge Node で 120 台のノードの挙動をエミュレートするに留まっている．

オーバーレイ構築ツールキットとして，Overlay Weaver[12] が挙げられる．エミュレーション機能を用いることで，動作試験を行いながらオーバーレイネットワークの開発が可能になる．1 台の計算機で 4,000 ノードのエミュレーションが可能であることが確認されている．しかし，ノード間の通信において，通信遅延がエミュレートされることがないため，通信遅延を考慮したオーバーレイネットワークの評価に用いることができない．

インターネット上に分散配置したノードで構成された分散システムのテストベッドである PlanetLab[13] を用いれば約 1,000 台の実機によるオーバーレイネットワークの評価が可能である．しかし，PlanetLab は実際のインターネットを用いたテストベッドであるため，ネットワークのバーストやリンクの切断といった特定の状況を意図的に発生させ，実験を行うことができない．

6 おわりに

本論文では通信の揺らぎを考慮したオーバーレイネットワークの開発支援環境として NetCamp を提案した．NetCamp はエンドホスト間の通信遅延のみを擬似的に再現することで，スケーラビリティを確保する．また遅延を与える機構をモジュール化し，開発者がそのモジュールを自由に作成することで，様々なネットワークの状況を想定し，オーバーレイネットワークを評価することができる．Network Coordinate System である GNP と Vivaldi を NetCamp 上に実装し，NetCamp が通信遅延を考慮したオーバーレイネットワークの評価に有用であることを確認した．この実験では各 Network Coordinate System を提案した文献が示す特徴を評価することができた．その際，約 1,000 台のノードの挙動を高い精度を保ちながら再現することができた．

今後の課題として，他のオーバーレイネットワークとして，DHT 等を NetCamp 上に実装し，評価が可能であるかを検証する必要がある．また，様々な遅延モジュールを作成する必要がある．

実験で使用したインターネットホスト間の遅延情報には IRIS Project[22] の方々が収集したものを利用させていただきました．

参考文献

[1] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S. E., Eaton, P. R., Geels, D., Gummadi, R., Rhea, S. C., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B. Y.: OceanStore: An Architecture for Global-Scale Persistent Storage., *Proc. of ASPLOS*, pp. 190–201 (2000).

[2] Dilley, J., Maggs, B. M., Parikh, J., Prokop, H., Sitaraman, R. K. and Weihl, W. E.: Globally Distributed Content Delivery., *IEEE Internet Computing*, Vol. 6, No. 5, pp. 50–58 (2002).

[3] Gnutella: . http://gnutella.wego.com.

[4] Cohen, B.: Incentives Build Robustness in BitTorrent. (2003). http://bitconjurer.org/BitTorrent/bittorrentecon.pdf.

[5] Freedman, M. J., Freudenthal, E. and Mazières, D.: Democratizing Content Publication with Coral., *Proc. of Symp. on NSDI*, pp. 239–252 (2004).

[6] Freedman, M. J., Lakshminarayanan, K. and Mazières, D.: OASIS: Anycast for Any Service., *Proc. of Symp. on NSDI*, USENIX, pp. 129–142 (2006).

[7] Eugene Ng, T. S. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches., *Proc. of IEEE Infocom*, pp. 170–179 (2002).

[8] Dabek, F., Cox, R., Kaashoek, F. and Morris, R.: Vivaldi: A Decentralized Network Coordinate System., *Proc. of SIGCOMM*, pp. 15–26 (2004).

[9] Xu, Z., Tang, C. and Zhang, Z.: Building Topology-Aware Overlays Using Global Soft-Stat., *Proc. of ICDCS*, pp. 500–508 (2003).

[10] *ns-2*. http://www.isi.edu/nsnam/ns/.

[11] Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostic, D., Chase, J. S. and Becker, D.: Scalability and Accuracy in a Large-Scale Network Emulator., *Proc. of OSDI* (2002).

[12] 首藤一幸, 田中良夫, 関口智嗣: オーバレイ構築ツールキット Overlay Weaver., 情報処理学会論文誌: コンピューティングシステム, Vol. 47, No. SIG12 (ACS 15), pp. 358–367 (2006).

[13] Bavier, A. C., Bowman, M., Chun, B. N., Culler, D. E., Karlin, S., Muir, S., Peterson, L. L., Roscoe, T., Spalink, T. and Wawrzoniak, M.: Operating Systems Support for Planetary-Scale Network Services., *Proc. of Symp. on NSDI*, pp. 253–266 (2004).

[14] Balakrishnan, H., Kaashoek, M. F., Karger, D. R., Morris, R. and Stoica, I.: Looking up data in P2P systems., *Communications of the ACM*, Vol. 46, No. 2, pp. 43–48 (2003).

[15] Ledlie, J., Pietzuch, P. R. and Seltzer, M. I.: Stable and Accurate Network Coordinates., *ICDCS*, p. 74 (2006).

[16] *GT-ITM: Georgia Tech Internetwork Topology Models*. http://www.cc.gatech.edu/projects/gtitm/.

[17] Zegura, E. W., Calvert, K. L. and Bhattacharjee, S.: How to Model an Internetwork., *Proc. of IEEE Infocom*, pp. 594–602 (1996).

[18] Allman, M. and Paxson, V.: On estimating end-to-end network path properties, *Proc. of SIGCOMM*, pp. 263–274 (1999).

[19] Gummadi, K. P., Saroiu, S. and Gribble, S. D.: King: estimating latency between arbitrary internet end hosts, *Proc. of SIGCOMM Workshop on Internet measurment*, pp. 5–18 (2002).

[20] *PeerSim*. http://peersim.sourceforge.net/.

[21] *p2psim*. https://project-iris.net/p2psim/.

[22] *IRIS Project*. http://project-iris.net/.