システムソフトウェアと
オペレーティング・システム 90－21
（２００２．６．２８）

P2Pネットワークシミュレータの設計と実装

柳原 正[2] 岩井 将行[2] 徳田 英幸[1 2]
[1] 慶應義塾大学 環境情報学部 [2] 慶應義塾大学大学院 政策・メディア研究科

近年、P2Pネットワークに関する研究が増加しているにも関わらず、実験を行うための手法が統一的に確立されていない。理由の一つとして既存のシミュレータがP2Pネットワークの特徴であるアプリケーションレベル・ルーティングとTCP/IPレベル・ルーティングをサポートしていないため、シミュレーションによる実験が行えないことがある。本論文ではP2Pネットワークのアプリケーションレベルルーティングをサポートしたシミュレーションが行えるシミュレータを提案する。また、P2Pネットワークシミュレータのプロトタイプとして N3 Simulator を開発した。N3 Simulator を用いることで、P2Pネットワークの開発者は容易にP2Pネットワーク用のアルゴリズム及びシステムのシミュレーションが行える。

Designing and Implementing a Simulator For P2P Networks

Tadashi YANAGIHARA[2] Masayuki IWAI[2] Hideyuki TOKUDA[1 2]

[1]Faculty of Environmental Information, Keio University
[2]Graduate School of Media and Governance, Keio University

Despite the increase of research on P2P network, the methods to commit tests for evaluation is insufficient. Existing network simulators do not support application level routing, making simulations based on various parameters difficult. We present a simulation model with support for application level routing, and have implemented a prototype, the "N3 Simulator". With the N3 Simulator, users can simulate P2P networks using approximate values and obtain values similar to those obtained in real world experiments. Also, users can easily perform tuning with algorithms and systems built for P2P networks.

1 はじめに

高速ネットワークの普及により、ネットワークを有効利用するための新たなアプリケーションとしてPeer-to-Peer(以降、P2Pと呼ぶ)ネットワークを利用したアプリケーションが普及している。例えばP2Pネットワークを利用したファイル共有アプリケーションとしてNapster[1]及びGnutella[2]が挙げられる。

また、P2Pネットワークは現在研究分野としても注目を浴びている[3][4][5][6][7]。しかし、研究として活発であるにも関わらず、実験に適したプラットホームが整備されておらず、互いを比較した評価が行われていない。P2Pネットワークはインターネットなど、広域ネットワーク上で利用されることを想定しているため、実機による実験を行うことが難しい。シミュレーションで実験を行う手段があるが、P2Pネットワークの特徴である「アプリケーションレベル・ルーティング」を実現したシミュレータが必要である。また、アプリケーションレベル・ルーティングに影響を与えるノード上の資源の表現、そしてノードとノードのアルゴリズムを分離が可能であることがシミュレーション内において必要である。本論文内でアプリケーションレベル・ルーティングを実現したシミュレータを提案し、その設計及び実装について説明する。

本稿では第2節でP2Pネットワークの実験を行う際の問題点を指摘する。次に第3節では実験環境としてP2Pネットワークシミュレータを提案する。そして、第4節、第5節においてP2PネットワークシミュレータであるN3 Simulatorの設計と実装について述べる。第6節では、N3 Simulatorの関連研究について述べる。第7節では今後の課題について述べる。

2 P2Pネットワーク

P2Pネットワークは、Domain Name System[8]とは独立した名前解決の方法を所持する。これはネットワーク内のノードは互いと通信し合うことでホストの発見が可能なアーキテクチャを利用しているためである。また、このようなアーキテクチャにより、旧来のサーバクライアントモデルのように特定個所が機能しなくなったときにネットワーク全体が停止することがなくなり、耐故障性に優れ、スケーラビリティもある。

P2Pネットワークが既存のネットワークと異なる点として「アプリケーションレベル・ルーティング」という特徴が挙げられる。一般に利用されるTCP/IPレベル・ルーティングとは別に、アプリケーションレベルでもう一つのルーティングテーブルを作成し利用する。このルーティングテーブルを基に、他検索アルゴリズムや配置アルゴリズムを所持している。

図1にP2Pネットワークの概念図を示す。P2Pネットワークの基本的な動作手順は次のように説明できる：

1. Aが使用するルーティングテーブル上に登録されたBとCにクエリを送信する
2. Cはクエリと一致するデータを所持していないので、次にDにクエリを転送する。
3. Dにはクエリと一致したデータを所持していたのでクエリが一致したという情報をAに返す。

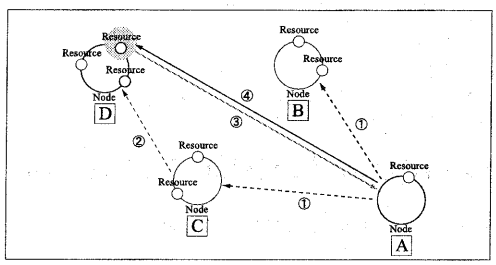

図1: P2Pネットワークの概念図

また、P2Pネットワークによってはノードが同種でも多様なものが存在する。例えばWinMX[9]ノードはNapsterプロトコルと独自のWPNPプロトコルを用いて他ノードと通信するため、Napsterネットワーク内では他のNapsterクライアントと同様、検索アルゴリズムはNapsterクライアントのものを使用するが、WinMXネットワーク内ではWinMXクライアントと同じ検索アルゴリズムを使用する。このように、ノードによって使用する検索アルゴリズムが異なることが想定されるため、ノード自身から検索アルゴリズムを分離することが望ましい。

2.1 P2Pネットワーク上の測定における問題点

P2Pネットワークにおける研究が近年活発である。例えば検索サービスのアルゴリズムを提供するChord[3]やCAN[4]、データへのアクセスを提供するOceanstore[5]やPAST[6]、Freenet[7]などがある。

しかし、研究として活発であるにも関わらず、アルゴリズムやシステムを測定するための手法は統一的に確立されていない。例えば、ホスト間通信を実行する際に必要とする通信量や時間などの値を求めるための測定環境を整えることが不可能であったり、既存のシミュレータ上で実行することが困難であったりためである。

P2Pネットワークのトラフィックなどの測定を行うための手法は二種類存在する。一つは複数台の実機を用いて計測する方法、もう一つはシミュレーションを通して計測する方法である。以下にこれらの二点について比較する。

複数の実機による実験

実測する場合では、複数のコンピュータでプライベートネットワークを組み、実測を行うことが一般的である。しかし、この実験環境では数十台の規模であれば実現は可能であるが、P2Pネットワークが想定するインターネットのような広帯域ネットワークでは、ネットワーク内のノード数が数百万台や数千万台になるため、実現することが非常に困難となる。

シミュレータによる実験

シミュレーションで測定を行う場合ではネットワークシミュレータの利用が可能である。代表的なネットワークシミュレータとしてNEST[10]、NESTから発展したREAL[11]、さらにREALから発展したns2[12]が挙げられる。

これらのネットワークシミュレータはTCP/IP通信におけるパラメータチューニングを目的として作られている。このため、通信における細かいパラメータ調整が行える。しかし、シミュレーション実行時に扱う要素が増加してしまうため、シミュレーションを行うための計算負荷が高くなってしまう。

さらにノードが利用する検索用及び配置用アルゴリズムがシステムによって異なることがあるため、アルゴリズムとノードを分離する必要がある。しかし、既存のネットワークシミュレータではノードを記述する際にノードとアルゴリズムを統合してプログラムするため、分離することができない。

3 P2Pネットワークシミュレータの概要

3.1 特徴

P2Pネットワークのアルゴリズム及びシステムの実験を行うためのプラットホームとしてP2Pネットワークシミュレータを提案する。既存のネットワークシミュレータと異なる点はノードとノードに適用する配置用及び検索用アルゴリズムを分離した点である。

これはノードが別々のアルゴリズムに従って動作するものが存在する場合にも有効であることがあげられる。例えばOceanStoreではCANを利用していたが、近年では新しくTapestry[13]を利用している。このようにP2Pネットワークを利用したアプリケーションによって配置用及び検索用アルゴリズムを場合によって切り替える必要がある。

アルゴリズムとノードの分離が可能なアーキテクチャにすることで、ノードとアルゴリズムの切り替えを行うことが可能となる。例えば開発者は作成したアプリケーションにChordを適応して実験を行った後、CANを適応して実験を行い、比較することができる。このため、アルゴリズムの

変更が用意となるため、実験が簡単に行える。

3.2 機能要件

P2P ネットワークシミュレータを構築する上では、以下の三点を考慮する。

ネットワーク内における各ノード上の資源の表現

P2P ネットワークのルーティング情報を決定する要因としてノード上のデータなどが考えられる。例えば Freenet では需要の高いデータを所持したノードへの通信を促進するようなアーキテクチャとなっている。このため、既存のネットワークシミュレータとは異なり、ノード上の資源の情報をシミュレーションの要素として含める必要がある。

検索アルゴリズムの多様な定義が可能なフレームワークの提供

P2P ネットワーク内のノードの検索及び配置アルゴリズムは多種多様である。例えば Chord で利用される配置アルゴリズムを適応したノードは、ネットワーク内に仮想的に円形状に並び、CAN で利用されるアルゴリズムを適応したノードは、ネットワーク内に仮想的に一つの大きな四角形を形成するように配置を行う。このように、様々なアルゴリズムを個別に定義し、シミュレータの利用が行えるようにしなければならない。

アプリケーションレベル・ルーティングのサポート

P2P ネットワークの特徴であるアプリケーションレベル・ルーティングによって、ノード間の TCP/IP 通信の各パラメータが決定するため、ネットワーク内の各ノードには二段のルーティングを行わなければならない。

- TCP/IP レベル・ルーティング
- アプリケーションレベル・ルーティング

アルゴリズムとノードの分離

上述したように、ノードに適用するアルゴリズムをノード自身から分離することでアルゴリズムの交換が簡単になり、さらにシミュレーションの計算量が少なくなる。特に P2P ネットワークのように広域ネットワークで構成される環境をシミュレートするにあたり、ネットワーク内のノードは多種多様となるため、これらの配慮は望ましい。

4 N3 Simulator の設計

本節では P2P ネットワークシミュレータのプロトタイプである N3 Simulator の設計について述べる。まず、本システムは大きく三つに分けられる。ネットワークのシミュレーションを行うためのプラットホームを提供する Engine 部、ノードがネットワーク内での挙動を定義する Driver 部、シミュレーション内にノードを生成するためのデータを記述した Node 部の三部である。図 2 にシステム構成図を示す。

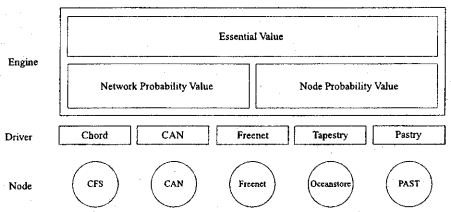

図 2: システム構成図

それぞれの部が提供する機能を次に記す。

4.1 Engine 部

Engine 部はシミュレーションを行うためのネットワークを作成する機能を果たす。シミュレーション開始時にユーザが定義したネットワークの規模やノード数を基にシミュレーションを行うためのネットワークを生成する。

また、シミュレーション内で不変である数値の多くはこの Engine 部に含まれる。プログラマの必要性に応じて拡張・変更することは可能であるが、基本的に変更を要するものは Driver 部で行われる。このアーキテクチャによってプログラマはシミュレータ内の乱数発生等のように、自動生成される数値などのプログラムを意識することなく利用できる。

Engine 部で定義される数値は大きく三つに分類できる：全シミュレーションが必要とする定数 (Essential Value)、ネットワーク全体に適応される確率依存な定数 (Network Probability Value)、ノード単位に適応される確率依存な定数 (Node Probability Value) である。それぞれについて、以下に詳しく記す。

Essential Value

全シミュレーションを通して利用される定数が存在する。例えば TCP/IP レベル TTL とアプリケーションレベル TTL、そしてノードを他ノードから区別するための ID の三つを考慮する。プログラマはここに属する定数を極力不変のままシミュレーションが行えるようにするため他の定数と分離した。また、シミュレーションを行う上で必ず設定しなければならない定数である。

Network Probability Value

確率によって値が決まる定数が存在するが、そのうちネットワーク全体に適用されるものがある。例えばノードの発生率などが挙げられる。現在はノードの生成率と削除率をサポートしている。

Node Probability Value

必ず全要素に共通していない数値も存在する。例えばノードが送信するクエリーの発生率はノードごとに異なる。このため、ノード別で利用される数値を分別する必要がある。現時点では各ノードからのクエリー送信の頻度を変更するように設計を行った。

4.2 Driver 部

Driver 部ではそのノードが所属するネットワークにおけるノード検索・配置アルゴリズムを定義する。ネットワークで定義されたノードを Driver 部に定義されたアルゴリズムは Engine 部に読み込まれ、シミュレーション内で利用されるネットワークに適用される。

また、Driver 部ではインタフェースという概念を提供する。インタフェースとはノードに適用されるアルゴリズムを抽象的に記述したテンプレートである。図 3 に Driver の概要を示す。

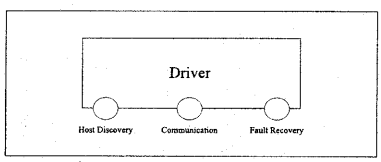

図 3: Driver の概念図

インタフェースは Host Discovery, Communication, Fault Recovery という三種類に分類される。それぞれについて以下に詳しく取り上げる。

Host Discovery Interface

Host Discovery を利用することでネットワーク内に新しいノードが接続したときのアルゴリズムを定義できる。しかし、P2P ネットワークを利用した既存のアプリケーションでは、新しいノードがネットワークに接続された際にノード間通信を通してルーティングテーブルを生成する方法を利用していない。その代わりに、キャッシュサーバを設置し、ネットワーク接続時に接続しキャッシュをダウンロードするようなアーキテクチャを取っているため、現時点のシステムでは Host Discovery のインタフェースは利用されることがない。しかし、現在キャッシュサーバを利用しない手法の開発は進めており、今後このような手法が実用化したときの対応のために準備をしておく。

Communication Interface

Communication はクエリを他ノードに送信するときに利用するインタフェースである。例えば Gnutella では所持するルーティングテーブル上の全ノードに対しクエリを送信するので表現は簡単であるが、PAST が利用する Pastry[14] では Domain Name System のような階層構造の関係を作成しているため、ツリー構造の関係を表記する必要がある。このように、ネットワーク内のノードにあわせたアルゴリズムに適した記述が行える。

Fault Recovery Interface

Fault Recovery はノードがネットワークから切断するときに利用するインタフェースである。ノードがネットワークから切断するとネットワークにノードが本来存在したはずの場所に穴ができてしまう。これが連続するとネットワーク全体のフラグメンテーションが発生し、通信効率が悪くなる。ここを定義することでフラグメンテーションが発生した際の修復が実行できる。

4.3 Node 部

Node 部では各ノードが所持する属性を定義する。属性とは例えばノードが所持するデータの種類や数、ノードが利用しているネットワークの通信速度などが挙げられる。ノードがネットワーク内で利用するアルゴリズムは Driver に記載する。Node 部で定義されたファイルは Engine 部に読み込み、ノードを生成する。

5 N3 Simulator の実装

本節では、第 4 節で述べた設計に基づいた N3 Simulator の実装について述べる。

実装は Windows2000 上で Java 言語を用いて行った。開発ツールは J2SDK1.4.0 を使用し、Engine 部のプロトタイプ、ノードを表現した Node 定義ファイル、そしてルーティングテーブル上の全ノードに対しブロードキャストで通信する構造が簡単な Driver 定義ファイルを実装した。これによって最も基本的な機能を提供する"Null Node"のシミュレーションが行える。

5.1 N3 Simulator のアーキテクチャ

N3 Simulator を構築する際に、シミュレータのアーキテクチャとして、NEST や ns2 と同様のシングルプロセス上で動作するように構築することができる。シングルプロセス上で動作させる利点として以下の点が挙げられる。

- ユーザが記述するプログラム (N3 Simulator では Node 部) の記述が簡単になる。
- シングルプロセス上で行うことでプロセス間通信を行う必要がなくなるため、シミュレータ自体のアーキテクチャが単純になる。

しかし、代わりに以上の点がしかし、シミュレーションの規模が大きくなった場合に以下の問題点が存在する。

- シミュレーションのノードを一斉で処理するため、特定のノードの集合を抽出してシミュレーションを行うのが困難となる。

- アプリケーション自体の負荷分散が行えない。

現在、N3 Simulatorはシングルプロセス上でシミュレーションを行っているが、今後シミュレーションの規模が大きくなることを考慮して、マルチプロセスに対応したモデルの方が望ましいと言える。なぜなら、ノードを個別でプロセス一つ上で実行ができるため、計算処理が向上し、さらに特定ノードの部分抽出も行えるためである。今後、N3 Simulatorの開発を進める上でマルチプロセスに対応したアーキテクチャへの移行も考慮している。

5.2 Engine部

Engine部はシミュレーションを行うプラットホームの基盤となる数値ジェネレータである。現在Engine部で利用可能な定数を表1に示す。これらの定数はNull Nodeのシミュレーションを行う上で最低限必要な定数であり、また他シミュレーションにおいても最も基本的な定数でもある。

Essential Value	
nodeID	ノードを区別するためのID
appTTL	アプリケーションレベルTTL
tcpTTL	TCP/IPレベルTTL
Unit Value	
querySize	クエリーのバイト数
Network Probability Value	
nodeCreateFrequency	新規ノードが生成される確率
nodeRemoveFrequency	新規ノードが削除される確率
Node Probability Value	
querySendFrequency	クエリーが送信される確率

表1: Engine内で利用される定数

5.3 Driver部

Null Nodeは他ノードに関するルーティングテーブルを所持し、クエリーを送信する際にこのルーティングテーブルに掲載されたノード全員へクエリーを送る。これはGnutellaノードで見られる挙動と同じである。このため、多段ハッシュテーブルを用意し、ネットワーク内における他ノードのIDをランダムで選び、ルーティングテーブルに記述する。これを基に、クエリーを送信するときにルーティングテーブルに表記された他ノードへクエリーを送信する。

5.4 Node部

Null Nodeはルーティングテーブルを所持するため、Node定義ファイルにはルーティングテーブルに関する情報を記述する必要がある。Null Nodeはルーティングテーブルに記す他ノードに関する情報として3~5ノード分のエントリを設けた。これは既存のGnutellaネットワークで一つのノードが所持するルーティングテーブルと近似的な値である。

5.5 動作手順

N3 Simulatorのプロトタイプの動作手順を今回実装したNull Nodeを用いて説明する。動作手順は図4に示す。

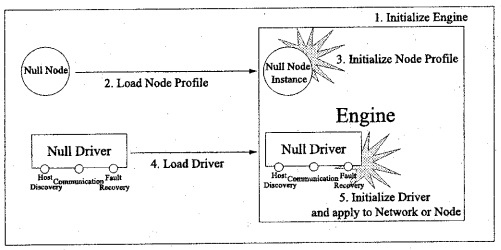

図4: Null Nodeを利用した動作手順図

まずシミュレータが起動されるとEngineが生成される。この段階ではEngine部はappTTLやtcpTTLなどの必要最低限の定数を生成する機能しか備わっていない。ここでNull NodeのNode定義ファイルがEngine部に読み込まれ、Null Nodeを生成する機能を得る。なお、この時点ではEngine部はノードが持つルーティングテーブルやファイルなど、様々な属性を持つことを認識しているが、ノードがどのようにネットワーク内で行動するかは認識していない。この例ではEngine部はNull Nodeはルーティングテーブルを所持していることを認識しているが、Null Nodeのルーティングテーブルに含まれている値で何をするかはまだ認識していない。

次にDriver定義ファイルがEngine部に読み込まれる。Driver定義ファイルにはネットワーク内で生成されるノードがどのようなアルゴリズムに基づいて行動するかが記されている。これを基に、Engine部は上記したノードをどのような配置及び検索アルゴリズムを利用しているのか、またそのノード上の属性をどのように扱うべきか、などが認識できる。この例ではNull Nodeのルーティングテーブル内に含まれたノードに対してクエリーを送信するように理解する。なお、Null Nodeはネットワーク接続時にGnutellaなどのように、キャッシュサーバなど、ネットワーク内の他ノード以外の方法でルーティングテーブルが生成し、そしてネットワーク切断時にネットワークフラグメンテーションなどの考慮を全く行っていないため、これらのインタフェースの表記はDriver定義ファイルに含まれていない。

6 関連研究

本節ではN3 Simulatorのプロトタイプシステムとの関連研究としてNEST、REAL、ns2を取り上げる。これらのシミュレータは互いを発展してき

たものであるため、まとめて"既存のネットワークシミュレータ"と呼ぶことにする。

既存のネットワークシミュレータはネットワークのシミュレーションを行うために利用可能なライブラリを提供する。利用者はC言語でこれらのライブラリを利用し、プログラムという形でノードを記述する。記述されたプログラムはシミュレータと共に一つのプロセス上で実行することでシミュレーションが行われる。しかし、このモデルの問題点としてアルゴリズムはノードを定義するプログラム内で表記するため、アルゴリズムに関する情報とノード自身の情報を切り放すことができない。ns2ではC++言語を利用しているため、ノードのモジュール化によって再利用が可能な部分が存在するが、ノードと配置及び検索アルゴリズムは同一のモジュール内に含んで記述するため、ノードを変更するためにはモジュールを一から記述し直さなければならない。

N3 SimulatorはJava言語によって記述されているため、ノード各部分のモジュール化が可能である。さらにNode部とDriver部によるノードとアルゴリズムの分離、そしてアルゴリズムにおいても三部に分離することでP2Pネットワーク内におけるノードの記述の表現性が高く、柔軟性及び拡張性が高いと言える。

7 おわりに

今後はN3 Simulatorのスケーラビリティなど、定量的評価が行えるまで開発を進める。また、今回構築したNull DriverはGnutellaノードと性質的に似ているが、Gnutellaはルーティングテーブルのノード間部分交換など、Null Driverで記載した内容以外の機能を備わっている。今後はGnutellaを含む、他システム用のDriver及びNodeを増やすように計画している。

モデル自身の問題点として、ネットワーク内のノードは全て同種であることを前提としているため、ネットワーク内に複数種類のノードが存在した場合には対応できない。例えばOpenNapネットワークのようにNapsterクライアントとNapsterサーバという二種類のノードネットワークがあるようなシミュレーションを行う際、現モデルではNapsterクライアントとNapsterサーバを区別することができない。この問題を対処するためにネットワークを多次元的に捕らえ、ノードを断層別に分別し、ノードを種類別で区別できるように考慮している。

本稿ではP2Pネットワークの実験環境が不十分である問題点を解決するために、アプリケーションレベルルーティングを表現したP2Pネットワークシミュレータを提案し、これを実現するためにN3 Simulatorの設計と実装を行った。本システムを利用することで、利用者は仮想値を利用したシミュレーションを通して実測値で行った実験と同様の実験結果を得ることができる。また、柔軟にシステム内で扱うノードの拡張が行えるため、今後新たに登場するP2Pネットワークのシミュレーションに対応できる。

謝辞

本研究を進めるにあたり、慶應義塾大学 徳田研究室の皆様に多大な御助言、御協力を頂きました。ここに深く感謝の意を表したいと思います。

参考文献

[1] Napster: "Napster". http://www.napster.com.

[2] Gnutella: "Gnutella". http://gnutella.wego.com.

[3] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H.: "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", *Proceeding of ACM SIGCOMM*, pp. 149–160 (2001).

[4] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: "A Scalable Content-Addressable Network", *Proceeding of ACM SIGCOMM* (2001).

[5] Kubiatowicz, J., Binzel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, . H., Weimer, W., Wells, C. and Zhao, B.: "OceanStore: An Architecture for Global-Scale Persistent Storage", *Proceeding of ASPLOS* (2000).

[6] Druschel, P. and Rowstron, A.: "PAST: A large-scale, persistent peer-to-peer storage utility", *Proceeding of HotOS VIII* (2001).

[7] Clarke, I., Sandberg, O., Wiley, B. and Hong, T. W.: "Freenet: A Distributed Anonymous Information Storage and Retrieval System", *Designing Issues in Anonymity and Unobservability* (2001).

[8] Group, N. W.: "Domain Name Structure and Delegation" (1994).

[9] WinMX: "WinMX". http://www.winmx.com.

[10] Dupuy, A., Schwartz, J., Yemini, Y. and Bacon, D. F.: "NEST: A Network Simulation and Prototyping Testbed", *Communications of the ACM* (1990).

[11] Keshav, S.: "REAL: A network simulator", *UCB CS Technical Report* (1988).

[12] LBL, PARC, X., UCB and USC/ISI: "The Network Simulator - ns-2 ". http://www.isi.edu/nsnam/ns/.

[13] Zhao, B. Y., Kubiatowicz, J. D. and Joseph, A. D.: "Tapestry: An infrastructure for Fault-tolerant Wide-area Location and Routing", *U.C. Berkeley Technical Report* (2000).

[14] Rowstron, A. and Druschel, P.: "Pastry: Scalable, distributed object location and routing for large-scale, persistent peer-to-peer storage utility", *IFIP/ACM International Conference on Distributed Plarforms* (2001).