社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－DPS－129（3）
2006／11／16

DHT 実装特性評価用エミュレータの開発と エミュレーション精度の評価

神谷俊之　加藤大志
ＮＥＣ インターネットシステム研究所

従来, DHT の評価は, 主に理論値及びシミュレーション評価によって行われていた. しかし, DHT を用いてアプリケーションを開発する際には，ソフトウェア実装の通信効率や応答時間などを評価することが重要となる．しかし，DHT 実装の評価はネットワーク環境の再現と多数の独立なノードを必要とすることから大規模化な評価が困難であった．そこで我々は PC クラスタ上でネットワークとノードのエミュレーションを行える評価基盤を開発し，大規模な評価を実現した．本稿では，本評価基盤のパケットロス再現機能，シナリオによるノード動作のグルーピング機能について説明し，実ネットワークとエミュレーションの比較による精度評価の結果について報告する．

Development of a DHT emulator and evaluation of its accuracy

Toshiyuki KAMIYA and Daishi KATO
Internet Systems Research Laboratories, NEC Corporation

The evaluation of DHT is mainly based on algorithm analysis and simulations. When we try to use DHT as a platform for an application, we need not only theoretical evaluations but also practical ones, such as network bandwidth or response time. But, a large scale evaluation of the implementation was difficult because a lot of independent node was required. We developed an emulator on a PC cluster to realize a large scale emulation. In this paper, we describe a network packet loss function and node grouping function, and show the comparison between the real nodes and the emulated ones.

1. はじめに

P2P (Peer-to-Peer)ネットワークの分野では，スケーラブルで効率のよいデータベースとして DHT (Distributed Hash Table, 分散ハッシュテーブル)技術が注目されている. DHT はすべて対等なノードで構成されたネットワークで擬似的な分散データベースを提供する. この疑似的なデータベースは，put(key, value)でデータを登録し，get(key)でデータを取得する．このデータの登録や取得は, どのノードで実行しても同じ結果を得ることができ, これが DHT の特徴となっている. 過去の研究において，DHT の様々なアルゴリズムやその実装が開発されてきた. 著名なものとして， Chord[10]， Tapestry[12]， Pastry[9]， Kademlia[7]などがある．これらの DHT アルゴリズムでは, 通信量や応答時間について理論的なオーダーの算出や, シミュレーションによる計測が行われている.

これに対し，我々は，DHT を利用したアプリケーション開発には実装レベルで性能評価できることが重要であると考え，PC クラスタを用いたネットワークエミュレーション上でノードの動作を制御する評価基盤を開発している. 著者らの以前の研究[15]では，ノードとネットワークの基本的なエミュレーションと, シナリオに基づく各ノードの動作記述, ログ取得が可能なシステムを構築した．しかし，本システムでは，動作シナリオとして各ノードの動作が均一である必要があるネットワークのパケットロスが記述できないなど実用的な環境を再現する上での制限があった. これに対し, 今回, 機能の追加拡張を行い, より柔軟なシナリオ記述を可能とすると共に, エミュレーション精度の検証を行ったので報告する.

以下，2 節で DHT の評価手法と我々の開発している評価基盤について概説し，3，4 節でシナリオ作成ツール，ネットワークエミュレーション機能の拡張について説明し，5 節で実ノードでの動作と比較したエミュレーションの精度について述べる．6 節でまとめと今後の課題について述べる．

2. DHT 性能評価法と評価基盤

2.1. DHT の実用性能評価手法

DHT を P2P 型のアプリケーション開発のための基盤として利用する際には，データ登録・検索の際の応答時間や，必要とするネットワーク帯域，各ノードでの CPU・メモリといった必要なリソース量の見積りが重要となる．これは例えば Web アプリケーション開発において，サーバの応答速度を検証したり，サーバに必要なリソースを検証したりするのと同様である．

しかし，DHT の場合，a)多数のノードが独立に動作し，各ノードの性能やノード間のネットワークの状態が均一であることは仮定できない．b)DHT では各ノードは少数のノードの情報のみを保持するだけで，ネットワーク内の任意ノードに到達可能であることを特徴とするが，近傍ノードについては，お互いに直接のリンクを保持する場合が多いため，正常な動作を確認するためには一定サイズ以上のネットワークが必要となる．これらの理由から DHT では，Web サーバと異なり多数ノード全体の挙動を評価する必要があるという独自の課題がある．

これに対し，従来，DHT の評価手法としては，p2psim[5]のようにシミュレーションにより評価する手法や，MACEDON[13]のように仮想コードによる評価が提案されている．また，実際のネットワークで実装を評価している例として OpenDHT[14]がある．しかし，シミュレーション評価は実際に実装されているコードとは違う疑似的なコードで動かすため、動作の正確な再現ができないという制約がある．また，MACEDON では独自の専用言語で記述されたコードをシミュレーション評価することができ，かつ，そのコードがそのまま実装コードとしても利用できる．しかし，任意の言語で記述された既存の DHT 実装を比較評価するのには適さない．実際のネットワークを用いる手法の場合にも，同一条件を再現することが困難で，異なる実装や同一実装の改良前後の比較が行えないという課題がある．

そこで我々は，任意の言語での DHT 実装の性能を比較評価し，また，これに基づいて実装を改良することを可能とするエミュレーションによる DHT 評価基盤を開発している．本評価基盤では，ノードのエミュレーション，ネットワークのエミュレーションにより，既存の DHT 実装をそのまま評価することを可能である．また，ネットワークの設定，各ノードの動作をシナリオ化し動作させることで，容易に同一条件での繰り返し実験，比較実験を行えることを特長としている．
以下では，本評価基盤の主要な 3 つの機能，ノードのエミュレーション，ネットワークのエミュレーション，評価のシナリオ化とログ収集・解析に関して概要を説明する．

1)ノードエミュレーション　1 台の物理的な PC(評価 PC)で複数のノードを動作させる機能である．これは，各ノードを異なるプロセスとして動作させることと，IP Aliasing を使って仮想的にネットワークを分けることで実現される．この方法で基本的には，通常の利用を想定した DHT 実装をそのまま動作させることができる．

ただし，次の 2 つの例外がある．IP Aliasing を使うと INADDR_ANY での bind はできなくなるため，そのような DHT 実装は仮想 IP アドレスに bind するように修正が必要となる．また，Java で実装された DHT は，JVM のメモリ消費を抑えるために 1 つの JVM 内で複数のスレッドとしてノードを動作させるように修正した方が効率的な場合がある．

2)ネットワークのエミュレーション　実際のネットワークでの遅延の影響を調べるため，任意のノード間の通信にパケット遅延を発生させる機能である．これは，静的なパケット遅延モデルに従って，Linux の tc (traffic control)を用い，カーネルレベルで遅延を発生させることで実現される．パケット遅延モデルでは，任意の IP アドレス間に任意の一定遅延を指定できる．この遅延モデルには，全ノード間で一定とするモデルや，p2psim[5]で使われている kingdata[3]を用いたモデルを利用可能である．

3)評価のシナリオ化とログ収集解析　本基盤では評価試験を開始する前にすべてのノードの動作をシナリオとして記述する．これにより，異なる DHT を同一のシナリオで評価したり，同一のシナリオで複数回の再現試験をしたりできる．シナ

リオには各ノードの動作として, 次の 4 つのイベントが記述される.
・join: ノードのネットワークへの参加
・leave: ノードのネットワークからの離脱
・put: ノードによるデータの登録
・get: ノードによるデータの取得
これらを時系列に並べたものがシナリオになる.

多数のノードの動作を個別に作成することは手間がかかるため, join/leave の平均間隔, 分布や, put/get の間隔など与えてシナリオを自動的に生成するシナリオ生成ツールを用意している.

これにより管理用の PC(操作 PC)で各ノードのシナリオを一括生成し, 各評価 PC に自動配布することができる.

また, 各ノードの動作は, ログとして出力され, 実験終了後, 操作 PC に集められ, GET 成功率, 応答時間, 平均通信帯域などを算出する. また, 評価 PC 単位で, CPU 使用率, メモリ空き容量も測定可能である. 評価 PC で動作させるノード数はシナリオ生成時に PC ごとに設定可能であるため, 特定の操作 PC について, 1 ノードだけを動かすようにすることで, 実働環境でのメモリ使用量, CPU 負荷を測定することが可能である. 本評価基盤の構成を図 1に示す.

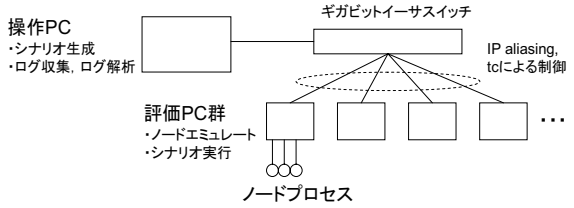

図 1 DHT 評価基盤の構成

2.2. 評価基盤の課題

著者らの以前の研究[15]ではネットワーク構成, ピア分布を再現可能で, シナリオによる動作記述が可能であったが, シナリオ記述できる動作に以下のような制限があった.

・全ノードが均質なシナリオのみ生成可能

[15]では, 対象とする P2P のシステムとして, 全てのノードが対等な関係のシステム, 例えば, 一般ユーザ間のファイル共有のような場合のみシナリオ自動生成が可能であった. しかし, コンテンツ配信のような場合, 特定のノード (配信者側のノード) のみが join/leave をほとんど行わないといった場合が想定されるが, このような動作を記述できなかった.

・ノードが保持するデータの指定ができない

実際のアプリケーションではノードに保持するデータ(value)の大きさはアプリケーションごとに異なる. 保持するデータが単純な数値なのか長い文字列なのか等は, 利用する帯域に影響するが, [15]ではシステムが生成するランダムなデータが自動的に割り当てられる.

・検索 key はランダム順序のみ生成可能

検索キー(key)の発生パターンもアプリケーションによって異なるが, put したデータに対してランダムな順序で検索を行うシナリオのみが生成可能であった.

・ネットワークのパケットロスに未対応

ネットワークのエミュレーションではパケット遅延のみが再現可能で, パケットロスの再現はできない.

今回, 我々はシナリオ生成ツールとネットワークエミュレーション機能を拡張することでこれらの課題を解決した. 以降の節でこれらの改良について説明する.

3. シナリオ生成ツールの機能拡張

シナリオ生成ツールでは, 利用者がノード全体の動作に関してパラメータを与えることで, 個別のノードの時間軸での動作をシナリオとして生成する. 今回, 機能拡張を行ったのはノードのグループ化に関連する機能, データの指定に関する機能である.

3.1. ノードグループ機能

今までのシナリオ生成では, シナリオパラメータが全ノードで共通でノードの動作は均一なものとなっていた. そこで, グループという概念を導入し, 不均一な動作を再現できるようにした.
グループにはそれぞれ個別のシナリオパラメータを指定することができる. これによって, 例えば, 次のようなシナリオを作ることができる.
・一部のノードだけが join/leave する
・一部のノードが頻繁に join/leave し, 他のノードはあまり join/leave しない
・一部のノードがデータを登録し, 他のノードがそのデータを取得する
・一部のノードがデータの登録取得に関与し, 他のノードは何もしない
グループのサイズは, ノード数または全体のサイズに対しての割合で指定する.

3.2. データの指定機能

今までDHTに登録するデータは基盤側でランダムに生成していた．しかし，DHT の評価をする際に, どのようなデータを登録するか予測できる場合は, そのデータを使って評価することが望ましい．基盤が生成するデータは key と value があるが．key については，実際の検索キーのハッシュ値が用いられるものとし, サイズは一定であることを仮定する．これに対し，value はアプリケーションによって内容が異なることが想定される．但し，評価の際には，value の実際の中身は一致判定に使われるのみであり, 評価結果に影響しない．そのため，パラメータとしては value のサイズを指定可能とした．

一方，key に関しては，ディレクトリにおけるルート情報のようにアプリケーションのつくりによって特定の検索キーにのみアクセスが集中する場合などを再現できるよう, 次の 3 種類のモードを用意した．

従来と同じ「ランダム」，指定された key のリストを順に使う「逐次選択」，指定された key のリストから任意に使う「ランダム選択」である．key のモードは，put と get それぞれに指定できる．ノードが put する回数より指定された key のリストの長さの方が小さい場合は，put が同じ key で繰り返されるため, 複数 value が登録されることになる．

4. ネットワークエミュレーションの拡張

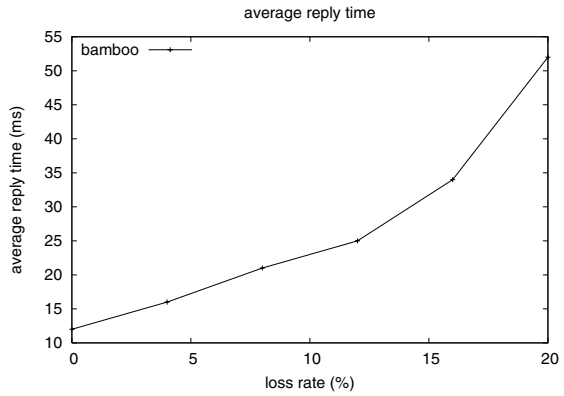

図 2 パケットロス測定結果

[15]では，ネットワークのエミュレーションとしてパケット遅延は実現していたが, パケットロスは実現できていなかった．すなわち，すべてのパケットが必ず宛先ノードに到達していた．今回，パケットロスを再現できるようにしたことにより, パケットが到達しない環境を想定してのより実際に近いネットワーク環境での評価ができるようになった．

評価基盤においてパケットロスが正常に機能していることを確認するため, パケットロス率を変化させた場合の get 応答時間を測定した．実験に使った DHT は Bamboo で，ノード数を 20 とし, ノードが一度ネットワークに参加したら二度と離脱しないタイプ(static)で実験を行った．
図 2は, パケットロスを 0%から 20%まで変化させたときのget応答時間の変化を示している．特別な関数関係があるわけではないが, パケットロスが明確にget応答時間に変化を与えていることから, パケットロスが正常に機能していると判断している．

5. エミュレーションの精度評価

[15]では，エミュレーションの精度評価として，同一パラメータ・同一乱数シード，同一パラメータ・異なる乱数シードで複数回測定を行うことで，測定ごとのばらつきを評価した．同一乱数シードの場合は各実験はまったく同じシナリオ, 異なる乱数シードの場合はノードの平均生存時間等は同一であるが異なるシナリオが生成される．それぞれについて 5 回測定を行った結果, 同一乱数シードの場合，各回の測定値のずれは 0.1%未満，異なる乱数シードの場合にも，通信量で 10%未満，応答時間で 5%未満のずれとなることを確認した．

しかし, この評価はあくまでエミュレーション自体の安定性に関する評価であり, エミュレーションが実働環境をうまくエミュレーションしているかについては未確認であった．そこで，ノードのエミュレーションおよびネットワークのエミュレーションの実際の環境とのずれを確認するための実験を行った．

5.1. ノードのエミュレーションの精度評価

ノードのエミュレーション動作の影響を調べるため, 実ノード環境とエミュレーション環境で同一シナリオを実行し結果を比較する．実験の対象とした DHT は Bamboo で，実働環境と同じく 1 つのノードをそれぞれ別の PC で動作させる場合(Npc)と，エミュレーション環境で複数のノードを一つの PC で動作させる場合(1pc)とを比較した．前者の場合は，ノード数分の PC が必要となるため大規模な実験はできず，ノード数は 11,

15，19，23 の 4 種類とした．ノードが一度ネットワークに参加したら二度と離脱しないタイプ(static)とノードが参加と離脱を繰り返すタイプ(join/leave)の 2 つについて評価を行った．

図 3, 図 4に実験結果を示す. get応答時間(図 3)は，static, join/leaveいずれのタイプでも，1pcとNpcの 2 つの場合の間は 5%程度の差に収まっている．総通信量(図 4)については，1%程度の差であり, 実環境に近い値が得られることが確認できた.

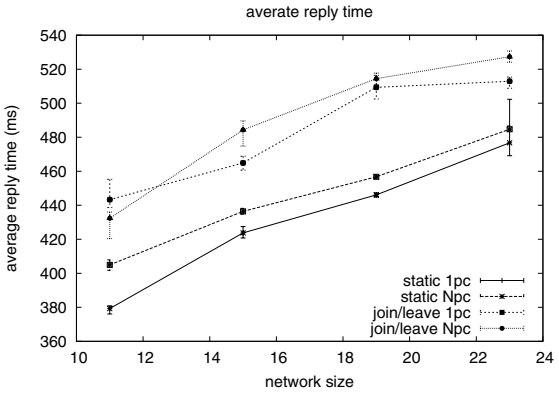

図 3　get 応答時間の比較

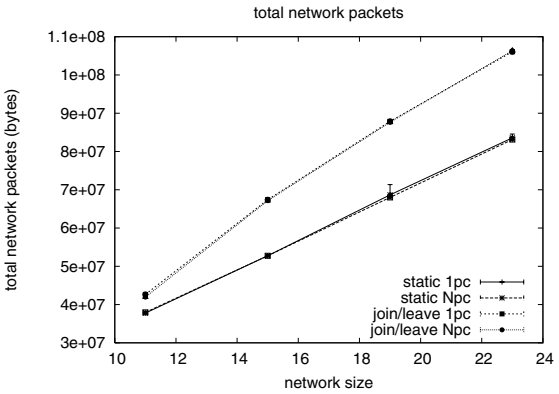

図 4　総通信量の比較

5.2. ネットワークエミュレーションの精度評価

ネットワークのエミュレーションのうちパケット遅延の効果についても検証を行った. 実験の対象としたDHTはBambooとChordである．ノード数を 20 とし，ノードが一度ネットワークに参加したら二度と離脱しないタイプ(static)において, すべてのIPアドレス間に一定の遅延を設定した場合の, get応答時間の変化を測定した. 図 5は, パケット遅延を 0msから 400msまで変化させたときのget応答時間の変化を示している．ノード数が 20 であるから平均ホップ数は,

$$\frac{1}{2}\log_2(20) = 2.16$$

であることが期待される. Chord は反復問い合わせを行うため，パケット送信回数の期待値は 2 倍の 4.32 になり，Bamboo は再帰問い合わせを行うため, パケット送信回数の期待値は 1 追加した 3.16 になる．パケット遅延と get 応答時間にはおよそ線形関係があることが読みとれ, その傾きもパケット送信回数の期待値に類似していることから, パケット遅延が正常に機能していると推測できる.

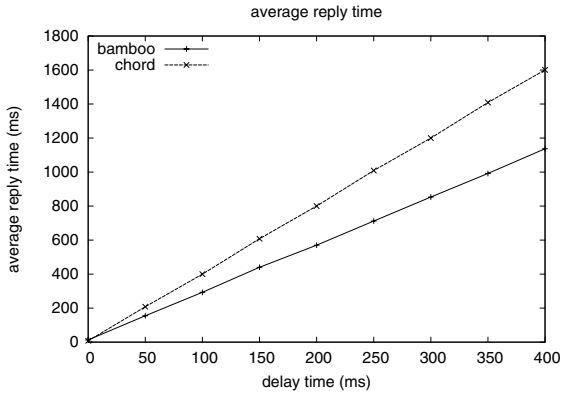

図 5　パケット遅延測定結果

6. まとめと今後の課題

従来, 我々の評価基盤では，ノードのエミュレーション，ネットワークのエミュレーション，評価のシナリオ化によって，3 つの DHT 実装(Chord, Bamboo, 独自実装である GISPv5)の基本的な性質については評価を行うことが可能であった．今回，エミュレーションの精度評価によりその精度を検証し，また，ネットワークのエミュレーションにおけるパケットロスの追加, シナリオ生成ツールの拡張を行ったことにより, 今後新たに実装される DHT を含め様々な DHT 実装をより実際の利用時に近い環境で評価することが可能となった.

以下では, 現状の基盤にまだ残されている課題を説明し，今後の方向性を述べる.

ノードのエミュレーションに関する課題としては, 本基盤では複数のノードが 1 つの PC を共有するため，ノード毎に細かいリソース(CPU, メモリ)制御ができないという問題がある．具体的には, あるノードが使用するリソースを制限することができないため, 処理能力の低いノードと処理能力の高いノードが混在したネットワーク

を再現することができない．これは，プロセスレベルでのエミュレーションの本質的な限界であり，これに対処するには仮想マシンなどの技術を導入する必要がある．

次に，ネットワークのエミュレーションに関する課題を説明する．既に述べたように，現状のネットワークのエミュレーションは，静的なモデルを再現しているのみで，動的なモデルには対応していない．具体的には，ネットワークの輻輳が再現できないという問題がある．我々は，実現が容易であるカーネルレベルでの制御方式を採用したため，静的なモデルのみを実現しているが，DHT の利用法が軽量な名前解決である限り，この方式で十分であると考えている．すなわち，輻輳が起きない程度の問い合わせで済むアプリケーションを想定する場合は，静的なモデルのエミュレーションで問題ない．しかし，アプリケーションとして輻輳が起こる場合や，DHT のみでもデータ量が増え輻輳が起こる可能性があれば，動的モデルの再現が必要となってくる．その場合は，動的なモデルのエミュレーションが必要となり，Modelnet[11]などの手法を用いるべきである．

最後に，シナリオ生成ツールに関する課題を説明する．シナリオ生成には様々な拡張が考えられ，今後も必要に応じて改良する必要がでてくると考えられる．現時点での一つの注目すべき問題は，ノードの join にゲートウエイノードを仮定していることである．これは，実験としては簡易でよいものの，実運用システムとしてはあまり成り立つことのない仮定である．これを解決するためには，ノードが join する際に，既に join しているノードの中からランダムか何からの指標によりノードを選び，そのノードを介してネットワークに参加するようにする必要がある．

7. おわりに

本稿では，我々が開発している DHT 実装をエミュレーションにより評価する基盤に関して，その概要と課題を述べ，いくつかの課題を解決するため行った機能の追加，拡張の詳細を述べた．また，精度の評価を示し，実環境と大きな差がないことを示した．これらの評価や改良により，さらに実際的な DHT のエミュレーション評価を行うことが可能となった．

参考文献

[1] http://freepastry.org/.

[2] http://overlayweaver.sourceforge.net/.

[3] http://pdos.csail.mit.edu/p2psim/kingdata/.

[4] http://www.bittorrent.org/.

[5] Jinyang Li et al. Comparing the performance of distributed hash tables under churn. In Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04), 2004.

[6] Jinyang L et al. Bandwidth efficient management of DHT routing tables. In the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05), May 2005.

[7] P. Maymounkov et al. A peer-to-peer information system based on the xor metric, 2002.

[8] Sean Rhea et al. Handling Churn in a DHT. In Proceedings of the 2004 USENIX Technical Conference, Boston, MA, USA, June 2004.

[9] A. Rowstron et al. Pastry: Scalable, decentralized object location and routing for largescale peer-to-peer systems. In Proc. Middleware 2001, 2001.

[10] Ion Stoica et al. Chord: A scalable Peer-To-Peer lookup service for internet applications.In Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160, 2001.

[11] A. Vahdat et al. Scalability and accuracy in a largescale network emulator, 2002.

[12] B. Y. Zhao et al. Tapestry: A resilient global-scale overlay for service deployment. Journal on selected area in communications, Vol. 22, No. 1, pp. 41–53, 2004.

[13] A. Rodriguez et al. MACEDON: Methodology for Automatically Creating, Evaluating and Designing Overlay Networks",. In Proc. of the 1st USENIX/ ACM Symposium on Networked Systems Design and Implementation (NSDI), 2004.

[14] Sean Rhea et al. OpenDHT: A Public DHT Service and Its Uses, Proceedings of ACM SIGCOMM 2005, 2005.

[15] 加藤大志，神谷俊之．DHT 性能評価法の提案と評価基盤の構築．情報処理学会研究報告，2006-DSM-40, pp. 31–36, 2006.