P2P を用いた RSS フィードでの一貫性の保持の向上

2007MI010 青木　康裕　2007MI105 小寺　英爾
指導教員　河野　浩之

1　はじめに

近年, web 上は不規則かつ頻繁に更新される傾向があり, 最新情報の効率的な入手が困難になるといった問題が発生している．この様な問題の解消として人気がある方法で RSS フィードが存在するが, 人気の高い RSS フィードに対する問い合わせが集中し, 帯域を無駄に消費してしまうといった欠点がある．

本研究では, P2P を用いた RSS フィードに対して一貫性の保持の向上を目指す．一貫性の保持の向上をおこなう対象として, P2P-RSS システムを用いたツールである FeedTree を選択する．この P2P-RSS システムの一貫性の保持の向上をおこなうために, 私たちは DHT アルゴリズムの get 成功率や length 数を比較し, P2P-RSS システムに用いられている DHT アルゴリズムに対して, より適した DHT アルゴリズムを提案することで一貫性の保持の向上を目指す．

以後，2 章では P2P システムを用いた RSS 技術の先行研究を紹介する．3 章では，P2P-RSS システムに適した DHT を提案する．4 章では，OverlayWeaver を用いた DHT のエミュレーション実験を行う．5 章では, 4 章で得たデータの比較検証を行う．最後に，結論と今後の課題で 6 章を締めくくる．

2　P2P を用いた RSS 技術の先行研究

本節では, 過去に行われた関連研究について, 実際に P2P を RSS に用いた FeedTree[1] また, FeedTree で用いられている P2P アルゴリズムである DHT[2] について紹介し, 本研究の方向性を議論する．

2.1　FeedTree

2.1.1　FeedTree のアルゴリズム

FeedTree では RSS 等の新着記事の情報を Scribe を用いて利用者へ発信する．Scribe は Pastry 等の P2P オーバーレイネットワーク上に組み立てられたスケーラブルなグループ通信手法である．Scribe の特徴として, 木構造の中で子がいないことを検出するための周期的なメカニズムを用いていてこのメカニズムが軽量的なものであり, メッセージが無い時だけ呼び出される．このメカニズムは, グループの数に比例して大きくなることを示している．これらの特性は FeedTree の様な大規模イベント通知システムを構築するのに適しているアルゴリズムだといえる．

しかし, 先行研究において Scribe はサイトの更新情報となる RSS を得るために巡回するのを目的とした FeedTree の購読システムの設計に対して適していなかったと判断された．本研究では FeedTree に適したアルゴリズムを提案し, より一貫性の保持を向上させることを目的とする．

2.1.2　FeedTree の構造

FeedTree は，図 1[*1] のように構築されている．プラグインモジュールと JDOM XML ライブラリに依存する ROME ツールキットは, 分析して生成した API を WebProxy と Publisher に提供する．FreePastry はオーバレイネットワークと Scribe Multicast/anycast サービスをコアに提供する．BC(BouncyCastle) crypt は暗号プリミティブをコアに提供する．

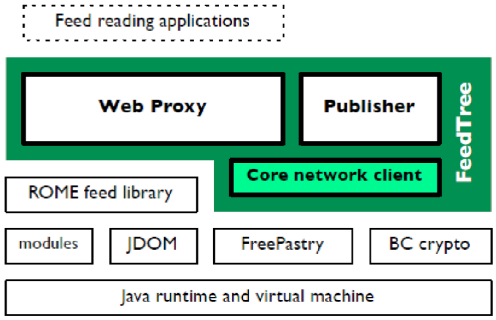

図 1　FeedTree の構造

2.2　DHT

DHT とは, Distributed Hash Table の略でハッシュテーブルを複数のピアで管理する技術のことであり, FeedTree にも用いられている．DHT を用いることにより, Pure-P2P であってもネットワーク負荷を抑えて, ネットワークに参加している全 PC との通信が可能になる．また, DHT を用いると相手の PC に対する検索も非常に高速である．Pure-P2P の場合は数 10 万ノード位が限界だと言われているが, DHT を用いることで数 10 億ノードの通信が可能となる．このことから DHT は FeedTree の様な大規模なイベント通知システムを構築するのに適していると考えられる．

[*1] FeedTree, http://www.feedtree.net/

表 1　DHT アルゴリズム比較表

	CAN	Chord	Tapestry	Pastry	Kademlia
アーキテクチャ	多面体構造	円状構造	プラクストン構造	プラクストン構造	XOR 論理式
検索プロトコル	key・座標	key・NodeID	NodeID の suffix	NodeID の prefix	key・NodeID
ルーティングパフォーマンス	$O(d.N^{1/d})$	$O(logN)$	$O(log_BN)$	$O(log_BN)$	$O(log_BN)$
ピア参加・離脱	2d	$(logN)^2$	log_BN	log_BN	log_BN

本研究では，FeedTree で使用されている FreePastry と他の DHT アルゴリズムを比較し，より一貫性の保持に適した DHT アルゴリズムを提案する．ここで, DHT アルゴリズムである CAN, Chord, Tapestry, Pastry, Kademlia 等の特徴を表 1 で紹介する．

3　P2P-RSS システムに適した DHT の提案

本章では, 一貫性の保持の向上を行う P2P-RSS システムを提案する上で, DHT に求められる機能について調査を行い, エミュレーション等を用いて比較・検証を行う項目について議論をする．

3.1　P2P-RSS システムの比較項目

本節では P2P-RSS という観点から一貫性の保持を検証するための比較項目を選出する議論を行う．本研究における一貫性では, ノード間における更新時間のずれの減少を目指す．

3.2　FeedTree の欠点

FeedTree に用いられている Scribe では, ヘテロジニアスな環境において様々な問題が発生する．(ここでのヘテロジニアスな環境とは OS や機器が異なる環境やネットワークであることを指す.) そのひとつとして, 帯域幅の制限によって発生する pushdown が構築される木の性能に対して大きな影響を与えるというものがある．

一般的には, ノードの上り帯域には制限がかかっているので, 各ノードがこのマルチキャスト木上で保持することが可能とされる子ノード数には限界がある．この保持することが可能な子ノード数の最大値を Degree と定義する．Scribe のシステムではマルチキャスト木構築時に Degree 以上の join 要求を受けた場合, pushdown と呼ばれる操作を行い, 子ノード数を Degree 以下に抑える仕組みを保持している． この pushdown 手法は Scribe 上では二種類あり, その内のひとつである Preempt Degree pushdown 手法はヘテロな環境でも高さの低いマルチキャスト木を構築可能とされている．

しかし, Preempt Degree pushdown 手法は Degree のみしか考慮していないために, 親ノードからどんなに離れているノードであろうとも, Degree さえ大きければ pushdown 実行時に優先される. その結果, 親子ノード間の距離が大きいマルチキャスト木が構築されてしまう場合があり, ネットワーク資源の消費が大きくなるという問題点がある. これは大規模なイベント通知システムに対して不適なものである．

3.3　ツール選定

本節では, 前節で議論した get 成功率, 同時接続可能ピア数を比較項目とし, シミュレーション・エミュレーション可能なツールを紹介・選定する．候補となるツールを表 2 にあげる．

表 2　使用ツール比較

OverlayWaeaver	NS-2	OMNeT++
J2SE を持つ OS	Linux	Windows, Linux
Java	C++,OTcl	C++
複数の DHT アルゴリズムを実装 自作アルゴリズムの実相が容易に可能 アルゴリズムの変更が容易に可能 PC1 台で数十万ノードを生成して実験可能 エミュレーションの設定が容易	パケットの転送経路を確認可能 パケット転送の様子を視覚化可能 さまざまなネットワーク上の機能を実装	NED という高級言語を使用可能 GUI を用いてパラメータを設定可能 実験中はパラメータの変更不可 シミュレーションの拡張が可能

3.4　使用するソフトウェア

本研究において OverlayWeaver をエミュレーションを行うソフトウェアとして選択する．OverlayWeaver を選択した理由として 複数の DHT アルゴリズムを選択可能なであり, さらにアルゴリズムのみを差し替え公正な比較が可能であるため, 本研究における私達が測定したい値を計測することに適しているという点が挙げられる．また, OMNeT++ において本ソフトウェアの利点であるシミュレーションの視覚化機能を用いた場合, 私達が計測する際に設定するノード数の条件を満たすことができず, 本研究における私達が測定したい値を測定することが不可能であったという点が挙げられる．

4　P2P-RSS システムに適した DHT の比較

本章では, エミュレーションを行う実験環境の紹介と, エミュレーションを用いて P2P-RSS システムに適した DHT について比較・評価を行う.

4.1　実験環境

エミュレーションは以下のような環境で行う.

- PC スペック
  - OS:Windows 7 Ultimate 64bit
  - メモリ:6GB
  - CPU:Intel(R) Core(TM)i7 CPU 920 @2.67GHz
- 使用ツール
  - OverlayWeaver 0.9.11
  - eclipse 3.6 Helios

4.2　エミュレーション内容

本節では, 本研究で行うエミュレーション内容について説明する. メッセージロスト率はノード間での通信時に起きる伝達失敗の可能性のことである. よってノード間での通信回数が多いほどメッセージロストは生じやすい. すなわち, あるノードが必要としているノードを探索し, データを get するまでにかかる経路数である length 数が多い場合ほどメッセージロストによるタイムアウトが起きる可能性が高くなり, 一貫性の保持に影響を与える可能性が高まってしまうと考えられる. メッセージロスト率が高いほど, パケットを正しく得ることができなくなることから, 情報の get 成功率が低下すると考える. これを検証するために get 成功率と length 数を Pastry, Tapestry, Chord, Kademlia の計 4 パターンの計測を行う. また, 1000 ノードから 1000 刻みで計 5 パターンの計測を行うものとする.

4.3　エミュレーションの手順

本節ではネットワークエミュレータを用いて, 各 DHT の比較エミュレーションを行う. 今回は各ノードのメッセージロスト率を変更した際の get 成功率, ノード数を変更した際の length 数についての比較エミュレーションを行う. エミュレーションにおける条件として, ノード数, エミュレーション比較で用いるアルゴリズム, メッセージロスト率を以下のように設定する.

表 3　シナリオ項目

ノード数	1000,2000,3000,4000,5000
アルゴリズム	Pastry, Chord, Tapestry, Kademlia
メッセージロスト率	0,1,2,3,4,5 %

計測の手順と方法は以下のようにして行う.

1. 設定した値のネットワークを作成
2. put コマンドにより RSS データに見立てた値を DHT に格納
3. get コマンドを使用し各ノードの到達値を計測
4. status コマンドを使用し各ノードに値が到達するまでの length を計測
5. get 成功率, length の値からアルゴリズムの比較

5　エミュレーション結果の検証

本章では, 4 章で行ったエミュレーション実験結果の検証を行い, P2P-RSS システムの一貫性の保持の向上に最適なアルゴリズムの選択を行う.

5.1　各アルゴリズムの get 成功率

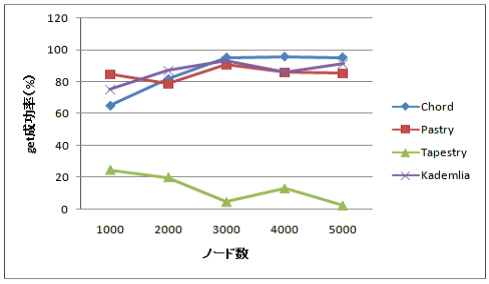

図 2　get 成功率の推移

表 4　get 成功数

ノード数	1000	2000	3000	4000	5000
Chord	652	1636	2857	3831	4767
Pastry	848	1579	2722	3447	4266
Tapestry	246	339	134	516	106
Kademlia	752	1744	2792	3436	4567

エミュレーションでは, メッセージロスト率を 0%, 1%, 2%, 3%, 4%, 5% と変更した環境の中でノード数 1000 ノード, 2000 ノード, 3000 ノード, 4000 ノード, 5000 ノードに対してアルゴリズムを変更した際の get 成功率の計測を行う. エミュレーションの結果, Pastry, Kademlia が全てのノード数において安定して高い get 成功率を出した. 各アルゴリズム間の get 成功率の変動が顕著であったメッセージロスト率 5% 時の 1000 ノードから 5000 ノードの場合における各 get 成功量の値の変動を図 2, 表 4 に示す. 横軸はノード数で, 縦軸は get 成功率である. また, 各アルゴリズムにおける考察を以下に示す.

5.2 各アルゴリズムの length 数の分布

本節では, 各アルゴリズムの get を行った際の length 数の分布の比較を行う. 横軸は length 数で, 縦軸はノード数である. エミュレーションの結果, Pastry, Tapestry が安定した length 数を出した. 各アルゴリズムにおける length 数のばらつきが顕著であったノード数 4000 時の計測結果を図 3 に示す.

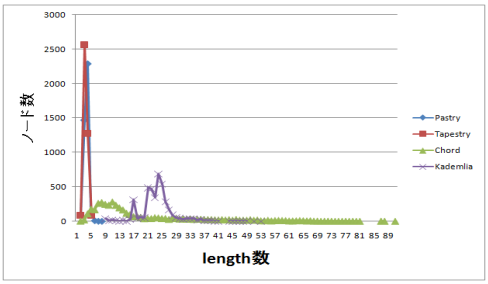

図 3 length 数の分布

5.3 length 数と get 成功率の関係についての考察

本節では, get 成功率, length 数の分布の関連から各アルゴリズムに対する考察を行なう. 5.1 節で比較的高い get 成功率を出した Pastry, Kademlia の length 数のノード数変化による変化を比較する. Pastry, Kademlia におけるノードを変化させた際の length 数の変化をまとめ図 4, 図 5 に表示する.

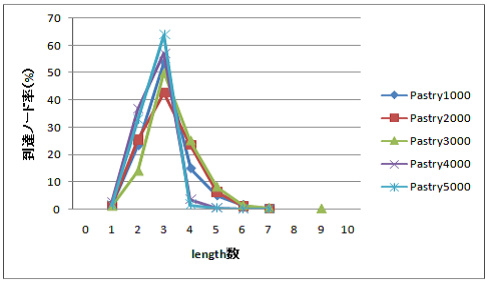

図 4 Pastry の length 数分布

Pastry では, 全てのノード数において length 数は 10 以下と低い値でまとまっている. またグラフの概形もノード数の変化にかかわらず安定して分布しているため, ノード数を変更しても安定した length 数を保持できると考えられる. それに対して, Kademlia では, グラフの概形はノード数の変化と関わらず安定しているが, length 数の分布は広くノード数が増えるにつれて, length 数は広く分布していくことがわかる.

本研究では, メッセージロスト環境下での get 成功率

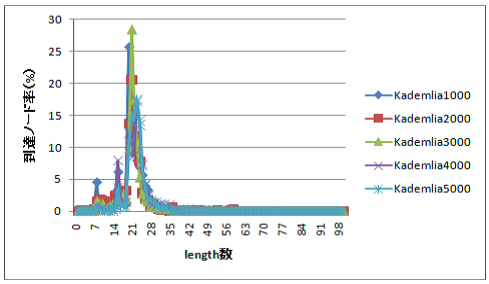

図 5 Kademlia の length 数分布

や length 数のエミュレーションを行い, P2P を用いた RSS フィードにおいてより一貫性の保持に適した DHT アルゴリズムの選択を行った. エミュレーション実験の結果, Pastry では最もメッセージロスト率の影響を受けた 5% の状態で, 1000 ノード時 84.8% , 2000 ノード時 78.95% , 3000 ノード時 90.73% , 4000 ノード時 86.18% , 5000 ノード時 85.32% と多くの値で安定して平均より高い数値を出している. また, length 数が 10 以下といった低い値でまとまり, ノード数が増えても概形に乱れがなく安定していることから, Pastry が最も P2P-RSS システムの一貫性の保持に適したアルゴリズムであると考えられる.

6 おわりに

本研究では, Pastry が最も P2P-RSS システムでの一貫性の保持の向上を行うための DHT アルゴリズムとして適していると選択を行った. 今後の課題として, 各アルゴリズムの Churn 耐性や, ノード消失時の影響, また, Pastry を利用したマルチキャスト手法に関して提案・検証を行う必要がある.

参考文献

[1] Dan Sandler, Alan Mislove, Ansley Post, Peter Druschel,“ FeedTree: Sharing Web micronews with peer-to-peer event notification, ”Peer-to-Peer Systems IV, pp.141-151, 2005.

[2] Eng Keibg Lua, Jon Crowcroft, Marcelo Pias, “ A Survey and comparison of Peer-to-Peer overlay network schemes, ”IEEE Communications Surveys and Tutorials, Vol.7, pp.72-93, 2005.

[3] 八重倉智, 松尾啓志, “ Scribe における効率的なマルチキャスト木を構築する pushdown 手法の提案, ”情報処理学会研究報告, UBI, [ユビキタスコンピューティングシステム], pp.73-78, 2006.