Peer-to-Peer アーキテクチャのメッセージ共有への応用

鳥居大祐 1)　板倉陽一郎 1)　田中裕一郎 2)　横澤誠 1), 3)　篠原健 1), 3)
1) 京都大学情報学研究科　2)ベリングポイント株式会社　3)野村総合研究所

近年、インターネットの利用者の爆発的普及、PC の高機能化、ブロードバンドの普及により、分散協調を旨とする Peer-to-Peer 方式が注目されている。Peer-to-Peer 方式は、クライアント･サーバ型と違い、ネットワークの負荷を分散させるだけでなく、新しい知識や情報の共有をサポートするシステム設計方法として期待されている。本研究では、この Peer-to-Peer アーキテクチャを応用したメッセージ共有システムを開発し、システムの実証実験を行い、動作確認を取ることができた。また、シミュレータを開発し、シミュレーションを行った結果、実証実験の測定結果をほぼ再現することができた。

Application of Peer-to-Peer Architecture to Message Sharing

Daisuke TORII 1) Yoichiro ITAKURA 1) Yuichiro TANAKA 2)
Makoto YOKOZAWA 1), 3) Takeshi SHINOHARA 1), 3)
1) Graduate school of Informatics, Kyoto University 2) Bearing Point Co., Ltd.
3) Nomura Research Institute, Ltd

Due to growing usage of broad-band network and high performance personal computers, Peer-to-Peer based distributed computing has drawn attention to the new system architecture. This architecture can distribute the concentrated resource usage of servers which are usually observed in client-server systems, and it will hopefully become a new architecture for the knowledge and information sharing. In this research, the message sharing design based on Peer-to-Peer architecture is developed and its effectiveness of this architecture is demonstrated by an actual experiment with users. Moreover, a separately developed Peer-to-Peer simulator well explained the mechanism of this message sharing system.

1.　はじめに

近年，Peer-to-Peer(以下、PtoP)アーキテクチャへの期待が高まっている。

PtoP アーキテクチャとは，クライアント-サーバーアーキテクチャ（C/S）の対となる概念であり，ネットワークに参加するそれぞれの PC（これを Peer と呼ぶ, 仲間の意。）が対等な関係を持ってネットワークを構成するかたちである。これに対してクライアント-サーバーアーキテクチャはサーバーが全てのサービスを提供し，クライアントがそれを利用するかたちをなす。C/S と PtoP アーキテクチャのネットワーク構造を図１に示す。

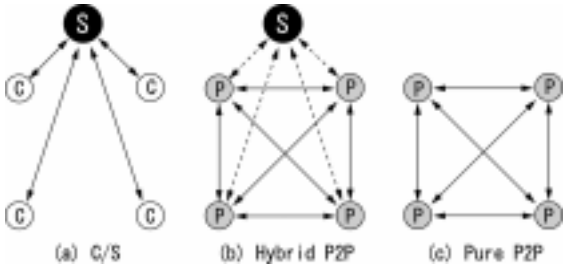

図１：C/S と P2P のネットワーク構造

クライアント-サーバーアーキテクチャでは，サーバーがサービスを提供し，クライアントがそれを利用する形を採るため，サーバーに負荷が集中する。さらに、ブロードバンド化によって個人の回線容量が増加し，更に高性能化により処理速度が上がったためサーバー側の回線容量が限界になることが想定されうる。実際、一日に数百万のアクセ

スがある巨大掲示板サイトなどでは，しばしばサーバーダウンが起きている。

そこで，サーバーの負担を減らし，個人の PC の高性能化を有効活用することが PtoP アーキテクチャへの要望の一つである。

2. PtoP アーキテクチャのメッセージ共有への応用

2.1. PtoP アーキテクチャへの産業界のニーズ

特にビジネスの方面から PtoP に期待されているのが, 知識共有，コラボレーションの面での貢献である。時々刻々と変化するビジネス環境において，知識共有，コラボレーションのために各人が統一したフォーマットに記入し，決められたサーバーにアップロードするという形は極めて非効率的である。更に，書式を統一しても企業・分野横断的な知識共有の際にはいちいち 1 対 1 で変換する必要が生じ, 結局リターンを勘案した場合知識共有へのインセンティブを失うことになるのがしばしばであった。産業界が求めるのは，たった今起きたビジネスの過程で生じたドキュメントをそのまま知識共有して活用できる環境であり，そこでは手元にある情報をそのまま（as-is）共有できるアーキテクチャが必要になる。このためのコラボレーションツールとしては PtoP では老舗である Groove [3]がよく知られている。

2.2. PtoP アーキテクチャの性能評価

このように, 知識共有に資する PtoP アーキテクチャの開発, 評価への需要は高まっていると言えるが，PtoP システムの開発, 評価には困難が伴う。というのも，PtoP ネットワークは，リアルタイムでネットワークの規模，形状が変化するので観察が困難という特徴を備えており，その結果評価実験には多くの人手が必要になり，しかも追試を伴うには大規模な施設が必要となってくる。Peer が自由に参加, 離脱することが PtoP の最大の特徴でありスケーラビリティを生み出している点でもあるが，ネットワーク全体を評価しようとする際には障害となってくる。数 1000 人規模の実験を複数回行うのは，Peer となるのが各自の PC であって管理しきれない以上，非現実的である。

そこで評価実験の手法としては,

(1) 実験参加者や期間を絞って実験設計し，データを得る。
(2) シミュレータによりネットワークを擬似的に構築し，設定したパラメータについて考察する。

という二通りが考えられる。(1)については様々なデータが得られる半面，追試は難しく、実験参加者や期間をどの程度に設定するかが問題となってくる。一方，(2)については実際に人手を用いては不可能な 1 万，10 万 Peer 規模でのシミュレーションが行える半面，設定したパラメータ以外については考慮しておらず，やり取りされるデータの意味的な側面を切り捨てることになるので，モデルの構築が問題となってくる。[4]

今回の研究においてはこの両方のアプローチを採り，実際に PtoP のメッセージ共有システムを実装した上で評価実験[1]と専用シミュレータを実装したうえでの評価を行った。

3. PtoP メッセージ共有システムの実装

3.1. ノードの構成要素

本研究で開発した PtoP メッセージ共有システムの内容は以下の通りである。

メッセージ共有の方法には次に挙げるように、いくつかの選択肢が考えられる。

(1) 全てのメッセージの複製を受け取るレプリケーションモデル
(2) 選択しておいた条件に合致するメッセージのみを受け取るフィルタリングモデル
(3) 複製は受け取らず，必要に応じて他の Peer のメッセージを参照するオンデマンドモデル

本研究では(1)レプリケーションモデルを採用した。

各 Peer はランダムで他の Peer にアクセスし，自分の手元に無いメッセージを見つけるとメッセージを複写して取り込む。この動作の繰り返しで，誰かが投稿したメッセージはいずれ全ての Peer に行き渡ることになる。結果的に, PtoP 上で掲示板が再現されることとなる。

システムは C 言語と Perl で実装され, Linux を動作プラットフォームとした。

Peer 上で動作するユニットをノードと呼び，各ノードは次の 3 ユニットで構成される（図 2）。

・P/V（Post/Viewer)
　ユーザからのメッセージの投稿, 自身のノードで保持されているメッセージの表示といった要求を扱うユーザインターフェースを提供する
・SYNC（SYNChronizer）:

ランダムな間隔で他ノードの MACK と接続を確立し, 自身のノードに保持されていないメッセージがあればそのメッセージの送信を要求する

・MACK（Messaging Auto Communication Kit）：
通常は他ノードの SYNC からの接続を待っている状態で, 接続の確立後は保持したメッセージのリストや要求されたメッセージを送信する

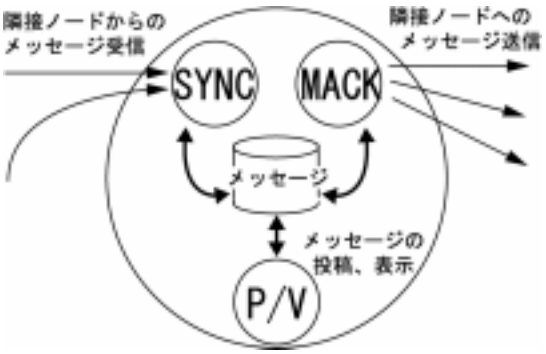

図 2：ノードの構成

3.2. ノード間の通信機能

メッセージの読み書きでは P/V が機能するが，ノード間のメッセージ同期作業は SYNC と MACK により行われる。

SYNC は一定の間隔ごとに他 Peer の MACK と接続を確立し，手元に無いメッセージの送信を要求する。

一方 MACK は SYNC からの接続を待ち，接続確立後はメッセージリストと要求されたメッセージを送信する（図 3）。

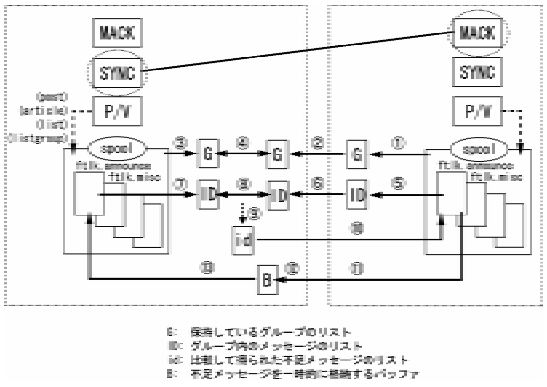

図 3：メッセージの同期作業

3.3. PtoP メッセージ共有の実験

実験の目的は, PtoP によるメッセージ共有の有効性を調べることにある。そこで,

- 完全性：メッセージが全てのユーザに行き渡るかどうか
- 健全性：メッセージの内容が破損しないかどうか
- 頑強性：大きな負荷が掛かっても動作するかどうか
- ユーザビリティ：利用しやすいシステムかどうか
- スケーラビリティ：Peer の増加に耐えられるかどうか

を判断基準とした。

被験者としては 50 人のユーザを集め, 同時参加を前提としてメッセージのやり取りをしてもらった。いずれも Web 上でのメッセージのやり取りに慣れたユーザである。メッセージ内容については全く無干渉である。書き込み（メッセージ送信）を行う部分は各人の Web ブラウザを用い，通信環境もそれぞれに任せてあるため，コントロールはしていない（図 4）。

メッセージの同期（SYNC 及び MACN）部分は 2 台の LINUX マシン上に、25 個のディレクトリを各ユーザのディスクスペースと見立てた環境を構築し，実験期間中常時稼動させておく形を採った。

実験期間は 2 週間であり, 実験中ユーザは Web ブラウザさえあれば好きなときに好きなだけメッセージが投稿できる。

図 4：Web ブラウザによる記事閲覧・投稿インターフェース

3.4. 解析方法

一定の書式に従った経路情報を解析することで，メッセージを受け取るまでに通った経路の数（ホップ数）などが

算出できる。ホップ数は、どれだけ効率的に記事が伝播したかという一つの指標とできる。

3.5. 実験結果

3.5.1. 計測データ

期間終了後，全ての SYNC と MACK のプロセスを終了して，メッセージの保存されているファイルを収集した。正味実験期間 12 日中で，50 人のユーザによって 278 件のメッセージが書き込まれた。

3.5.2. 結果

パスを分析することで，メッセージが通った平均経路数が得られた。最小は 1 ホップ，最大は 12 ホップであり，平均すると約 3.52 ホップで到達している（図 5）。この分布の分散は 2.68、標準偏差は 1.64 となった。

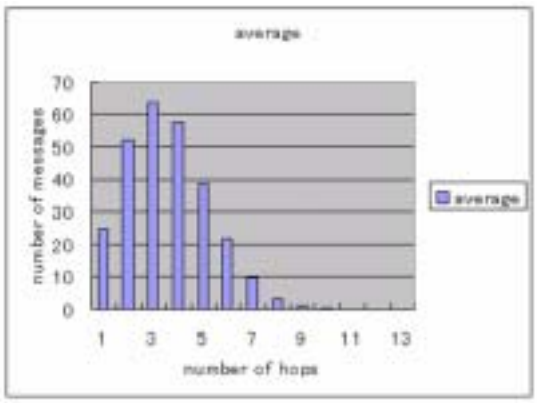

図 5：メッセージ共有に要したホップ数

3.5.3. アンケート結果

有効性を吟味するためにアンケートを行い，50 人中 35 人から有効な回答を得た。結果を図 6、図 7 に示す。

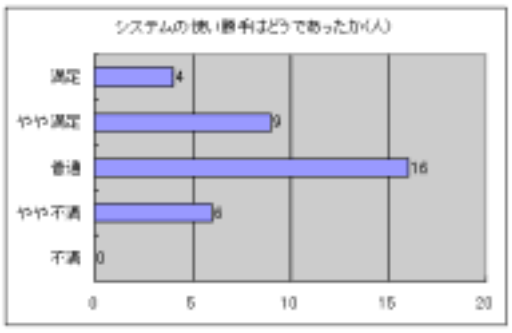

図 6：アンケート結果 1

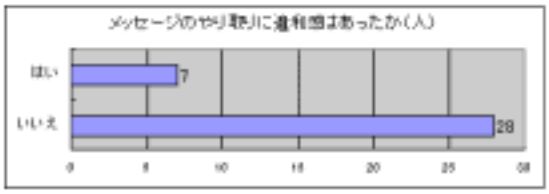

図 7：アンケート結果 2

3.5.4. 結果の検証

次に、3.3 で挙げたシステム評価の指標に結果を照らし合わせると次のようになる。

- 完全性：全てのメッセージはユーザに行き渡った。全てのユーザのスプールディレクトリには 278 件全てのメッセージが確認された。
- 健全性：書き込まれたメッセージの内容が破損することは無かった。
- 頑強性：同期処理を行っていたマシンは 25 のアドレスを管理するという高負荷を受けたが，支障をきたすことは無かった。又，ユーザはそれぞれ作業や仕事を行う傍ら実験に参加してもらったわけだが，特別システムに支障をきたしたという報告は無かった。
- ユーザビリティ：アンケート結果を勘案するに，さしたる不満は無いという結果が得られた。記述式回答では「レイアウトがシンプルすぎる」「レスの表示の工夫が欲しい」などの回答があったが，ユーザインターフェース部分の改善点である。
- スケーラビリティ：今回の実験では Peer の増加は行っていないが，少なくとも 50Peer では安定的に動作した。

4. PtoP メッセージ共有システム専用シミュレータの開発、検証

次に、2.3 で述べたように、シミュレータによる検証も行った。この章では、シミュレータの開発、また、その検証について述べる。

4.1. シミュレータ開発の目的

このシミュレータは、PtoP アーキテクチャにおけるメッセージ共有システムにおける完全性、スケーラビリティ、（頑強性）を試すものである。具体的には、記事の同期アルゴリズムの違いによって、記事がすべてのノードに行き渡るかどうか、またその効率はどうであるかということを

検証するものである。各ノードやマシンに対する負荷については、このシミュレータではそれほど細かく扱わない。記事同期の効率の指標としては、記事到達の平均ホップ数、平均到達時間などが考えられる。

4.2. シミュレータの設計

今回のシミュレータは、3 章で述べたシステムとほぼ同様の検証を行うものである。また、今後、別の記事同期アルゴリズムや実環境に合わせたパラメータを設定できるように、パラメータ設定部分と、動作アルゴリズム記述部分は切り離して設定できるようにする。システムの性質上、ある動作の次にはどういう動作をするかという状態変化を記述する事象中心型（離散事象型）のシミュレーションを行う。シミュレーション終了後、平均ホップ数と、ホップ数の分散を出力する。

4.3. シミュレータの実装

シミュレータの実装には、Java 言語（J2SDK-1.4.1）を用いた。オブジェクト指向型言語を用いることにより、ノードの数だけインスタンスを生成すればよく、リソースの管理が容易で、実環境に近い空間を想定できるためである。また、Java 言語は OS 間の移植が容易で、今後の拡張性に備える意味もある。

4.4. シミュレータによる実験

この実験では、3 章で述べたシステムとできるだけ同じ環境をパラメータにより表現して行った。パラメータとして調整したのは、記事同期プログラムである SYNC の起動時間間隔、ノード数、記事の総数である。具体値としては、起動時間間隔を 120 から 240 単位時間、ノード数を 50、記事の総数を 278 とした。

初期状態で、記事 278 件を、50 ノードに分散し、すべてのノードで同じ 278 件の記事を持つまで実行を行い、平均ホップ数とそのホップ数で到達したメッセージの数（50 人の平均値）を結果として得た。

シミュレーションは、10,000 回行い、それぞれのランでは乱数系統は別々に取り、それぞれで違う結果を得た。

4.5. シミュレータによる実験結果

10,000 回の実行すべてにおいて、記事は全ノードに到達した。また、10,000 回の平均値として、平均ホップ数は 3.39 となった。ホップ数ごとのメッセージの数を図８.に示す。分散は 2.34、標準偏差は 1.53 となった。

また、最大ホップ数の分布を図９に示す。最大ホップ数が 7 以下,15 以上となるのは極めて少ないことがわかる。

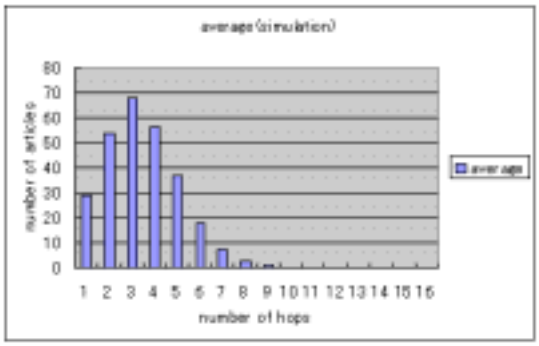

図８：シミュレーションによるホップ数の分散

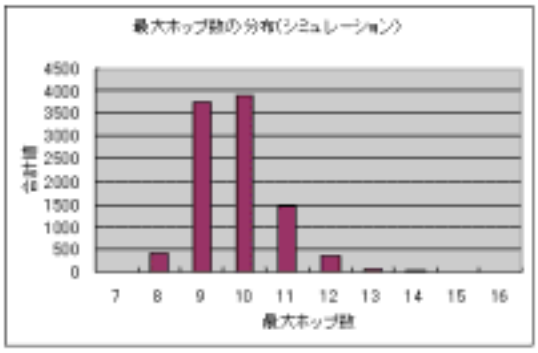

図９：シミュレーションによる最大ホップ数の分布

4.6. 結果の検証

3.3 で述べた評価実験とシミュレーションの平均ホップ数、分散、標準偏差の比較表を表１に示す。これから、平均ホップ数、分散に関して、それぞれの母平均、母分散に差がないという帰無仮説を設定し、t 検定を行った。これによると、平均ホップについては有意水準 35%において、分散については有意水準 15%で棄却された。

評価実験では、一台のマシン上に 25 ノードが仮想的に配置されており、一度にたくさんの処理が発生すると記事同期のパフォーマンスは低下した。また、記事はユーザにより任意の時刻に投稿された。これに対し、シミュレーションではマシンへの負荷は考慮されておらず、また記事は初期値として各ノードへ分散配置された。このようなパラメータの設定を今後シミュレーションに取り入れてやることが、シミュレーションの精度を上げることにつながると考える。

平均ホップ数、分散の帰無仮説は、有意水準 5%では採択されており、完全に妥当性を否定するものとはなっていないであろう。

	平均ホップ数	分散	標準偏差
評価実験	3.52	2.68	1.64
シミュレーション	3.39	2.34	1.53

表 1：評価実験とシミュレーションの比較

5. 考察

今回開発したメッセージ共有システムでは、3.5.2 で示したように、すべての記事はすべてのユーザへ到達し、また従来の掲示板と同様のユーザビリティが得られたことがわかり、P2P メッセージ共有システムの有効性が示されたと考える。ただし、記事の到達時間に関しては、C/S に比べて劣るのは明らかであり、動作メカニズムとして工夫が必要である。

シミュレータを使用した実験については、パラメータの選定とそれに対する結果への影響、その有効性をさらに検証する必要がある。例えば、ノードの On/Off、ネットワーク環境、各 Peer のマシン性能など考慮に入れてやらなくてはならないだろう。 今回のシミュレーションは、CPU が PentiumⅣ2.14GHz、メモリが 1G の PC を用いて行ったが、今後、ノード数の増加、同期アルゴリズムの変更に伴い、プログラムの最適化を行うほか、場合によってはスーパーコンピュータの使用も検討しなければならないと思われる。また、記事の同期アルゴリズムは、ランダムに一つのノードを選び、同期していくもので、これは数学的にもホップ数の期待値を計算できると考えられるので、今後は数学的にも考察していくつもりである。

平均ホップ数に関しては、評価実験とシミュレーションの値を比較してもさほど違いはなく、平均ホップ数はノード数と相関性があることも予測される。今後は、このような相関性についても検証していきたいと考える。また、ホップ数の評価に関しては，Stanley Milgram が行った Small-World Phenomenon の実験が参考になる。これは，「我々は皆少ない一連の知人関係で結ばれている」という現象であり, 1960 年代に次の様な実験を行ったうえで提唱された。アメリカ中西部から東海岸の全く面識が無い相手に対し，ファーストネームを呼び合う知り合いだけを経由して手紙を届けるという実験であり，ホップ数が約 6 であったという結果が得られている[5]。本研究では接続先はランダムに選ばれているが，スケーラビリティをアップし，接続先の選考に関しなんらかのパラメータを設定した場合（例えば, 今まで有用なメッセージを受け取った件数を「好感度」として設定する）に Small-World の結果に収束するような結果が得られれば，実験社会学的な成果になると考えられる。

6. おわりに

今回の研究では、PtoP メッセージ共有システムの有効性を、実システムとシミュレータの両方から検証した。実システムによる評価実験では、システムが問題なく動作することが確認され、シミュレータの実験では、評価実験をほぼ再現する結果が出た。

今後としては、このシステムの記事同期アルゴリズムを様々提案した上で、シミュレータによる検証を行い、メッセージ共有に適したシステムを開発していく。PtoP システムは、大人数で使われることを想定するべきであり、今後はシミュレータを用い、ノード数を増やして記事の到達やホップ数などを検証していくつもりである。また、PtoP 環境下で、単なるメッセージ共有ツールとしてだけでなく、新たなコラボレーションツール、グループウェアとしての可能性を模索していきたい。

参考文献

[1] **田中裕一郎 「Peer to Peerアーキテクチャのメッセージ共有への応用」 京都大学修士論文、2002年**

[2] **横澤誠「サーバーに依存しないPtoP型システムの設計」 野村総合研究所 知的資産創造 2002年9月号**

[3] Groove http://www.groove.net

[4] Joseph S. "*Adaptive Routing in Distributed Decentralized Systems: NeuroGrid, Gnutella and Freenet*", Proceedings of workshop on Infrastructure for Agents, MAS, and Scalable MAS, at Autonomous Agents 2001, Montreal, Canada

[5] Milgram, S. The small world problem, *Psychology Today* 1, 60-67, 1967.