分散データベース環境における P2P プロトコル経路制御手法

石田 常竹† 神尾 政和† 大室 学‡ 箱田 貴久†

†安川情報システム株式会社 〒215-0004 神奈川県川崎市麻生区万福寺 1-2-3 アーシスビル 5F
‡NTT アドバンステクノロジ株式会社 〒180-0006 東京都武蔵野市中町 1-19-3 武蔵野 YS ビル 5F
E-mail: †{isidat, kamio, hakoda}@ysknet.co.jp, ‡manabu.ohmuro@ntt-at.co.jp

あらまし 最近，Gnutella や Chord といった Peer-to-Peer (P2P)プロトコルが研究開発されてきている．P2P プロトコルは，P2P ネットワークに参加する各ノードが自身の隣人ノード(ネイバー)への要求伝達を繰り返すことで全体への要求を配送する．この仕組みは，(1) 全体のネットワーク構成を知らなくても要求の配送が可能である，(2) どのノードに目的データが存在するかを知っている必要がない，という特徴を持つ．しかし， P2P アプリケーションでは不要なトラヒックが通信の大半を占めてしまう問題がある．本論文では，この問題を解決しうる BONCHI プロトコルを提案する．

キーワード エージェント技術，分散コンピューティング，P2P

Peer-to-Peer Routing Protocol for Distributed Database Environment

Tsunetake ISHIDA† Masakazu KAMIO† Manabu OHMURO‡ Takahisa HAKODA†

†YASKAWA INFORMATION SYSTEMS Corporation 1-2-3 Manpuku-ji, Asao-ku, Kawasaki, 215-0004 Japan
‡NTT Advanced Technology Corporation 1-19-3 Nakamachi, Musashino-shi, Tokyo, 180-0006 Japan
E-mail: †hanako@denshi.ac.jp, ‡{taro, jiro}@jouhou.co.jp

Abstract Several protocols of Peer-to-Peer (P2P) systems such as Gnutella and Chord have been researched and developed in recent years. Each node in P2P network is cooperated with the exchanging messages to the next node (neighbor). P2P systems are featured that there is no necessity, (1) which a node recognizes topology and structure of P2P network, and (2) which node owns searched data in P2P network. However, nodes make generate a lot of messages in network. Here, we discuss routing protocol (named BONCHI protocol) how to reduce messages in P2P network compared with Gnutella protocol.

Keyword Agent technology, Distributed computing, Peer-to-Peer

1. はじめに

近年の Peer-to-Peer(P2P)プロトコルの研究開発によって，様々なアプリケーションで P2P プロトコルが実装されてきている（[4]，[9]）．P2P プロトコルによって実現されるサービスは，従来のクライアント/サーバ(C/S)モデルと異なり完全な分散システムであり，すべてのノードは等価で，サーバや階層が存在しない．P2P サービスに参加するノードは，ピア（peer）と呼ばれ，複数のピアから構成される論理ネットワークを構成する．1 つのピアは，複数のピアに接続でき，接続したピアのことをネイバー（neighbor）と呼ぶ．あるノードのサービス要求（例えば，ファイル検索）は，ネイバーに送信され，要求を受信したネイバーは，自身のネイバーへその要求を転送する．この転送を繰り返し，要求に応答できるピアを検索することが可能となる．このように，ピアから構成される P2P ネットワークでは，それぞれのピアが提供可能な資源を公開することにより，その資源をすべてのピアによって共有することが可能である．そのため，P2P サービスは，クライアント/サーバモデルで問題となっているサーバに対するネットワークと CPU，メモリといったノード自体の負荷を軽減することが可能である．また，分散したピアが自立的に P2P ネットワークに参加，離脱でき，1 つのピアに情報が集中しないことから，拡張性，耐障害性が高いという特徴を持つ．

現在までに様々な P2P プロトコルが提案されてきた．完全分散型ファイルシステムに応用されている Chord([1]，[5])やファイル共有アプリケーションとして応用されている Freenet([1]，[7])，Gnutella[8]等がある．しかしながら，Gnutella 等のように P2P ネットワーク全体にブロードキャスト通信するような P2P アプリケーションは，要求の転送時に経路を考慮していな

いことから，オーバヘッドが発生し，ネットワーク全体のメッセージ数が増加するという問題がある．我々は，上記問題に対する解決のための考え方を提案している[11]．本稿では，プロトコルが公開され，様々なソースコードが公開されている Gnutella をもとに，ネットワーク全体のメッセージ数を削減する BONCHI プロトコルを提案する．

本稿では，まず，2章で我々が用いるシステムモデルを示す． 3章では，我々の考えるメッセージ削減方法を提案し，4章で障害発生時の回避方法を示す．5章では，動的にネイバーを選択する方法を提案する．6章では，BONCHI プロトコルを実装したプロトタイプの性能評価結果より，有効性を示し，最後に7章でまとめと今後の課題を述べる．

2. システムモデル

2.1. P2P プロトコル

P2P プロトコルにおけるノード（ピア）は，ネイバーと呼ばれる別のノードからのメッセージを受信する．ノードは，受信したメッセージをメッセージの送信元ノード以外のすべてのネイバーへ転送する．送信元以外のネイバーが存在しない場合，転送を行わない．また，メッセージの到達する範囲を決定するために，様々な研究がなされているが([2], [6])，本プロトコルでは，TTL(time-to-live)を使用することを前提とする．P2P プロトコルには，要求に応答できるノードが発見できた場合に，転送を停止することで，全体のメッセージ数を削減するプロトコルも考えられているが，本稿では，転送を停止することなく，すべてのノードにメッセージが伝播できるブロードキャスト通信を前提とする．

本稿で対象とする Gnutella 等の既存の P2P プロトコルは，メッセージ転送を繰り返すことにより，全ノードへメッセージを伝達するブロードキャスト通信を実現する．しかし，P2P ネットワークの通信経路上にループ構造がある場合，このようなメッセージ転送の繰り返しは，通信の洪水を発生させる．Gnutella 等の P2P プロトコルでは，各ノードが重複したメッセージを破棄することでこの問題に対処している．

図 1は，Gnutella プロトコルが重複したメッセージを破棄する例である．図中の円は，ノードを示し，矢印は，ネイバーへのメッセージ転送を示す．ノード D がノード B からのメッセージ m_3 を受信後にノード C から m_4 を受信した場合，ノード D は，後から受信した m_4 を破棄する．このような重複したメッセージを破棄することで，ノード D がノード E へ同じメッセージを転送するのは，1 度のみとなる．

2.2. 冗長メッセージ

図 1において，ノード C はノード A からメッセージ m_2 を受信する度に，ノード D に m_4 を転送する．しかしながら，ノード D は，ノード B から受信する m_3 を受信している場合，同一メッセージである m_4 を必ず破棄する．同一メッセージを以下のように定義する．

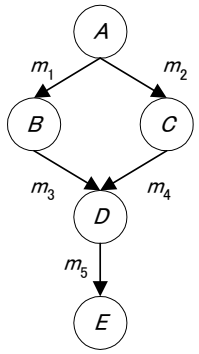


図 1　Gnutella プロトコル

【同一メッセージ】

同じノードから同じ時間に送信された同じ内容のメッセージをあるノードが複数受信した場合，そのメッセージは，そのノード上で同一メッセージである．□

本稿では，以下のメッセージを冗長メッセージグループと定義する．

【冗長メッセージグループ】

あるノードが同一メッセージを受信した場合，その同一メッセージの集合を冗長メッセージグループとする．□

プライマリメッセージと冗長メッセージをそれぞれ以下のよう定義する．

【プライマリメッセージ】

あるノード上で，受信したメッセージ m に対して，過去に同一メッセージを受信していない場合，m をプライマリメッセージとする．□

【冗長メッセージ】

あるノード上の冗長メッセージグループのうち，プライマリメッセージ以外のメッセージを冗長メッセージとする．□

図 1のノード D を例とすると，ノード D は， m_3 と m_4 という２つの同一メッセージから構成される冗長メッセージグループを保持している．冗長メッセージグループの内， m_4 を受信する前に m_3 を受信していた場合，m_3 は，プライマリメッセージ，m_4 は，冗長メッセージである．

本稿では，メッセージ転送時に毎回転送される同一メッセージの内，冗長メッセージの発生を抑制するこ

とで，既存の P2P プロトコルに比べ，トラフィック量を削減可能なプロトコルを提案する．

3. 冗長メッセージの削減方法

3.1. 転送停止手順1

本稿では，転送停止メッセージを導入することで，冗長メッセージを削減する方法を提案する．あるノードが冗長メッセージを受信した場合の手順(転送停止手順 1：図 2)を以下に示す．

1．　あるノードが冗長メッセージ(図中 m_4)を受信した場合，その冗長メッセージの送信先ノードに対して転送停止メッセージ(図中 o_1)を送信する(図中(a))．

2．　転送停止メッセージを受信したノードは，次回以降，そのノードに対してメッセージ転送を停止する(図中(b))．

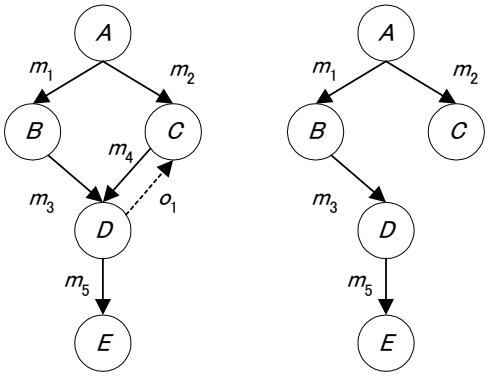


図 2　転送停止手順 1

転送停止手順 1 では，どのノードからそのメッセージが転送されてきたのかという転送経路情報を考慮していないため，あるメッセージは遠い経路を通る可能性がある．ここで言う遠い経路とは，転送するノード数が他の経路に比べ多いことを示す．図 3を例に問題点を示す．

図 3は，図 2の転送停止手順 1 の完了後に別のノード F から新しい要求である m_0 をノード C が受信した場合のメッセージの転送を示している．ノード C は，転送停止手順 1 によってノード D への転送を停止しているため，ノード A へメッセージを転送する．ここで，ノード D へのメッセージ転送数に着目する．転送停止手順 1 を実施しない場合，ノード D は，ノード C のネイバーであるため，ノード C は，ノード D へメッセージを転送する($F \rightarrow C \rightarrow D$)．転送停止手順 1 の実施後は，ノード C はノード A へメッセージを転送するため，遠い経路を通る($F \rightarrow C \rightarrow A \rightarrow B \rightarrow D$)．

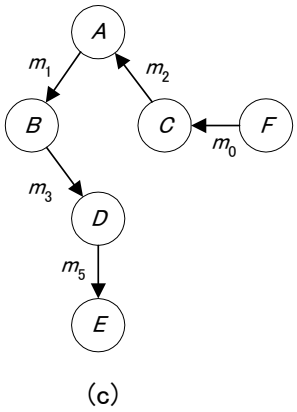


図 3　転送経路の考慮

本稿では，このような遠い経路を通ることを回避するために冗長パススタックを利用する．次に冗長パススタックと遠い経路を通ることの回避方法を示す．

3.2. 冗長パススタック

各ノードは，メッセージ転送時に以下に示すパススタックをメッセージに付加する．

【パススタック】

メッセージ m を送信したノードの IP アドレス，もしくは，FQDN N (ノード情報)の順序集合情報をパススタック $\mathrm{path}(m) = [N_0, N_1, \ldots, N_n]$とする．ここで，$N_0$ が先頭，N_n が最後のノード情報を示す．□

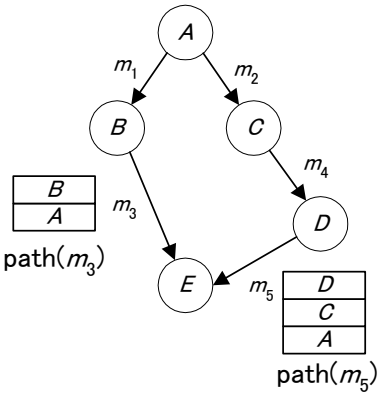
本プロトコルでは，メッセージの転送時，そのメッセージ内に自ノードの IP アドレス，もしくは，FQDN を順次追加していくことによりパススタックを実現する．パススタック情報から，どのノードを経由してきたかという経路を知ることが可能となる．以降，この経路のことをパスとする．

転送停止手順 1 では，ノード C はすべてのメッセージをノード D へ転送しないとしていた．すなわち，メッセージの送信先情報によってのみ，メッセージを転送するか否かを決定していた．遠い経路を回避するために，メッセージの送信先情報とメッセージがどの経路を通過したのかを受信したメッセージに付加されているパススタックと転送停止メッセージに付加されているパススタックを利用することで判断できる．

どの経路を通過したかどうかを判断するための情報である冗長パススタックをパススタックから生成する方法を示す．図 4は，ノード E がプライマリメッセージ m_3 と冗長メッセージ m_5 を受信した場合にノード E が冗長パススタックを作成する例である．図 4を使用して冗長パススタックの生成手順を示す．今後，以

下の手順で生成されたパススタックを冗長パススタックと呼ぶ.

1. ノード E は，m_3 を受信時に m_3 を保持しておく.
2. m_5 を受信時に，プライマリメッセージ m_3 のパススタック path(m_3) = [N_A, N_B]と冗長メッセージ m_5 のパススタック path(m_5) = [N_A, N_C, N_D]を取得する.
3. path(m_3)の最後のノード情報を(一番初めは N_B)を取り出す.
4. 取り出したノード情報を path(m_5)の最後から検索する.
   (ア) 見つからなかった場合は，手順３に戻り次のノード情報を取り出す.
   (イ) 見つかった場合は，path(m_5)の見つかった場所までのパスを取り出し，そのパススタックを冗長パススタックとし，処理を終了する.



冗長パススタックの生成方法

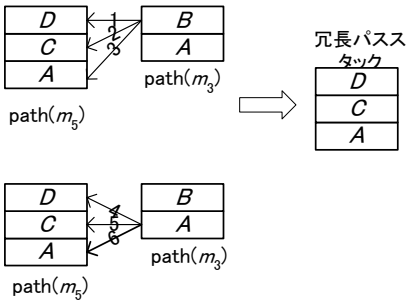


図 4 冗長パススタック

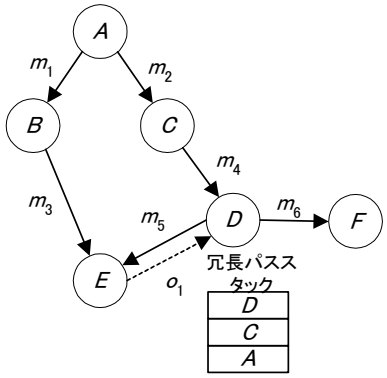
3.3. 転送停止手順 2

転送停止手順１と異なり，転送停止メッセージに冗長パススタックを追加した新たな手順を以下に示す(転送停止手順２：図 5(a)).

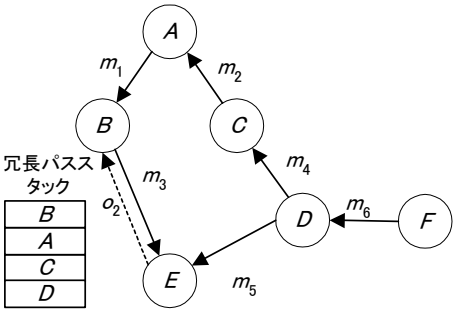
1. ノード E が冗長メッセージ(図中 m_5)を受信した場合，その冗長メッセージの送信先ノード D に対して，冗長パススタック path(o_1) = [N_A, N_C, N_D]を付加した転送停止メッセージ o_1 を送信する.
2. 冗長パススタックを含んだ転送停止メッセージを受信したノード D は，次回以降，次の条件を満たす場合のみ，ノード E に対してのメッセージ転送しない.

【条件】

受信したメッセージのパススタックの先頭がノード E から受信した冗長パススタックを含む場合は，メッセージを転送しない．□



(a) ノード*A*からのメッセージ送信



(b) ノード*F*からのメッセージ送信

図 5 例：転送停止手順２

例えば，図中ノード D は，パススタックが path(m_n) = [N_A, N_C, N_D]という m_n を受信した場合，そのメッセージをノード E へ転送しない.

次にノード F がメッセージを送信した場合を考える. 図 5(b)は，ノード E がノード B から冗長メッセージを受信した場合を示す．ノード E は，ノード B に対して冗長パススタック path(o_2) = [N_B, N_A, N_C, N_D]を送信す

る．最終的にノード D は，ノード A からのメッセージをノード E に転送しないが，ノード F からのメッセージはノード E に転送する．

このように，転送停止手順２を導入することにより，経路を考慮したメッセージの転送が可能となる．また，経路の決定に送信元のノード情報を利用する方法も考えられるが，BONCHI プロトコルでは，ループしている経路上のパススタックだけを記録していればよいので，送信元のノード情報を記憶する場合に比べ，記憶領域を小さくすることが可能である．

4. 障害回避

転送停止手順２によって，図 6(a)から図 6 (b)のようなメッセージ配送になった場合を考える．ここで，ノード B に障害が発生した時，ノード C は，従来，ノード D へメッセージを転送するはずであったが，転送停止手順２によって，メッセージを転送しないため，ノード D へメッセージを転送するノードがなくなってしまう(図 6(c))．

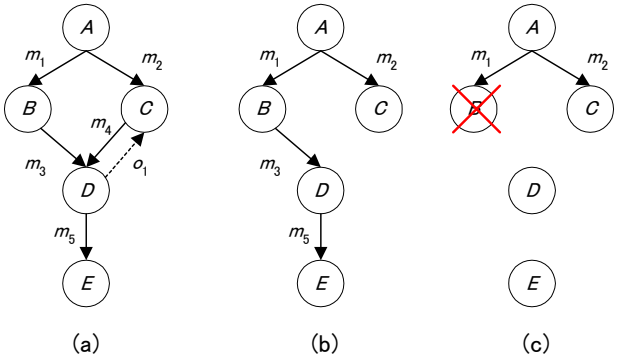


図 6　ノードに障害が発生した場合

この問題を解決するために，本プロトコルでは，ノードに障害が発生した場合に，(１)転送停止初期化メッセージを導入する方法と，(２)ネイバーの持つネイバー情報をもとにした方法の２方法を提案する．

4.1. 転送停止初期化メッセージによる障害回避方法

ここで，ノードは，ネイバーへメッセージ送信時に送信エラーを検知できると仮定する．また，ノードは，メッセージ送信の前に TCP における 3 ハンドシェイクのように送信前にネイバーとの通信経路を確立する機能を有しているものとする．図 7は，ノード A が，転送停止初期化メッセージを送信する例である．ノード A は，ノード B にメッセージを転送時に，送信エラーが発生する．このとき，ノード A がノード C へメッセージを送信するときに，送停止初期化フラグ r を付加した送停止初期化メッセージを送信する．ノード A は，ネイバーであるノード B とノード C への通信経路を確立した後に，メッセージを送信するものとすると，ノード A がノード B へメッセージを送信する前にノード C へメッセージを転送することはない．送停止初期化フラグ r を付加した送停止初期化メッセージを受信したノード C は，送信停止情報をすべて初期化する．そのため，ノード C は，ノード D へメッセージを転送する．

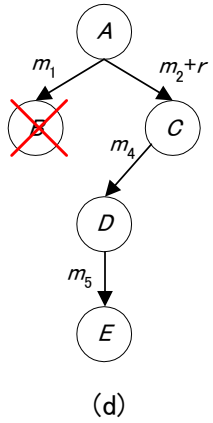


図 7　転送停止初期化メッセージの送信

4.2. ネイバー情報による障害回避方法

ここで，転送停止初期化メッセージではなく，ネイバー情報を利用した障害回避方法を提案する．ネイバー情報とは，あるノードがネイバーとしているノードの情報を示す．

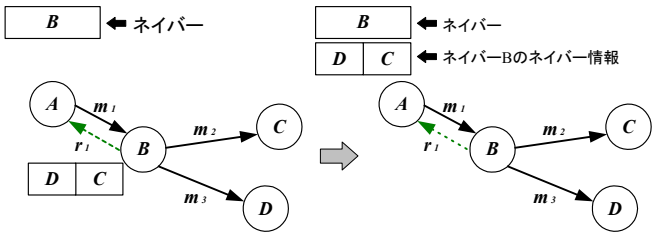


図 8　ネイバー情報の交換

図 8は，ノード A とノード B とのネイバー情報の交換を示す．ノード B は，ノード A からの要求を受信したときに，その応答として，自身のネイバー情報をノード A へ送信する．ノード B のネイバー情報を受信したノード A は，そのネイバー情報を受信し，保持する．

図 9を例に，ノード B に障害が発生した場合のノード A の処理を示す．ノード B の障害を検出したノード

A は，事前に入手していたノード *B* のネイバー情報からノード *B* のネイバーとしていたノードを自身の新しいネイバーとする．この処理によりノード *B* からメッセージを受信していたノード *C* とノード *D* は，それぞれ，ノード *A* から新たにメッセージを受信することが可能となる．

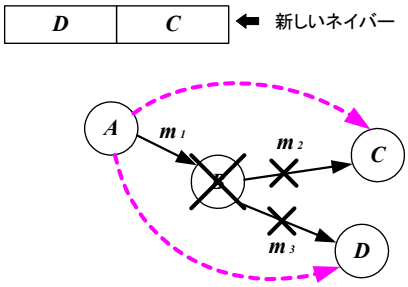


図 9　ネイバーの障害発生時の処理

4.3. 障害回避方法の比較

ここで，転送停止初期化メッセージによる障害回避方法とネイバー情報による障害回避方法を比較する．転送停止初期化メッセージを導入する場合，障害発生時に特殊なメッセージを送信する手順のため，障害回避のために余分なデータを持っている必要がない．しかし，図 10の例のようにノード *D* に障害が発生した場合，ノード *C* に転送停止初期化メッセージが到達しないために，ノード *E* にメッセージが到達しないといった問題がある．一方，ネイバー情報による障害回避方法では，障害回避のために，ネイバー情報を保持していなければならないが，転送停止初期化メッセージの問題点のようにメッセージが到達しないということはない．

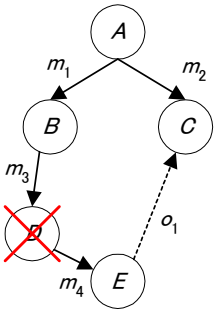


図 10　障害停止初期化メッセージの問題点

保持しなければならないデータが多くなったとしても，すべてのノードに確実にメッセージを伝達する必要のあるシステムは，ネイバー情報による障害回避方法を導入し，逆に，保持するデータを少なくする必要がある場合は，転送停止初期化メッセージによる障害回避方法を導入すればよいことがわかる．

5. ネイバーの動的選択方法

既存の P2P プロトコルでは，P2P ネットワークに参加したノードのネイバーは，固定される．一方，要求を複数送信し，応答を複数返すようなノードとの関係性は，非常に高いと考えることができる．例えば，論文ファイルを検索する要求を送信したノードに対して，あるノードが何度も応答を返すような場合，そのノードは，論文ファイルを多く持っていると考えられ，ノード間の関連性は高いと考えることができる．このように関連性の高いノードをネイバーと選択することで，メッセージ数の削減，検索速度の高速化を計ることができる．ネイバーの選択方法には，以下の方法が考えられる．

（１）　あるノードの応答の回数がしきい値を越えた場合，そのノードをネイバーとする．

（２）　応答を返した複数のノードのうち，ある確率でそのノードをネイバーとする．

（３）　上記，（１）と（２）を組み合わせる．

また，要求は，分類することが可能であると仮定し，ある分類の応答を返すノードをネイバーに切り替えることで，より効率のよい検索が可能となる．

例えば，ノード *C* に対して論文ファイルの検索要求に何度も応答を返すノード *A* と企業に関するファイルの検索要求に応答を返すノード *B* があった場合を考える．ノード *C* は，論文ファイルを検索する場合には，ノード *A* をネイバーし，企業に関するファイルを検索する場合は，ノード *B* をネイバーとすることで，より効率のよい検索が可能になると考えられる．

6. 評価

我々は，他の既存の P2P システムとの論理的なメッセージ数の比較を行う．更に，既存の Gnutella アプリケーションと BONCHI プロトコルのプロトタイプをメッセージ量という観点で性能評価した．既存の Gnutella アプリケーションには，Gnut[10]（バージョン: 0.4.28）を使用した．本評価では，5章で示したノードの動的選択方法は，考慮しない．

本評価では、*n* 台のノードがそれぞれ *m* 台のネイバーを持つリング型ネットワークでのトラヒック量を考察する。図 11は，*n*=7，*m*=4 の場合の例である．ただし，ネイバーに関しては，ノード 1 のみを示している．*n*=7 の場合，*m*=6 の時が，フルメッシュとなる．

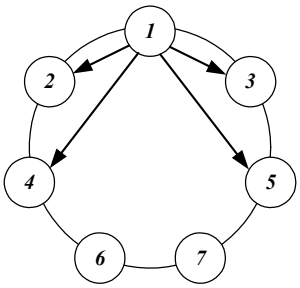


図 11　評価ネットワーク例

6.1. メッセージ数の論理的評価

ここで，Gnutella 等の既存の P2P システムと提案するプロトコルのメッセージ数の論理的評価を行う．n 台のノードがそれぞれ m 台のネイバーを持つ場合の要求数 Q と冗長メッセージ数 O は、それぞれ下式で求めることができる．ここで，条件として送信元のノードには，要求を送信しないものとする．

$$Q = n - 1$$

$$O = n \times m - 2 \times (n - 1)$$

従って、既存の P2P システムにおけるメッセージ数 M_1 は、下式から得られる。

$$M_1 = Q + O = n \times m - n + 1$$

一方、我々の提案するプロトコルの転送停止手順 2 の処理後のメッセージ数 M_2 は、下式から得られる。

$$M_2 = M_1 - O = Q = n - 1$$

M_1 の m は、n で表すことができ($m = n - k$ ($1 \leqq k \leqq n - 1$))，M_1 のオーダは $O(n^2)$，M_2 のオーダは $O(n)$となる．

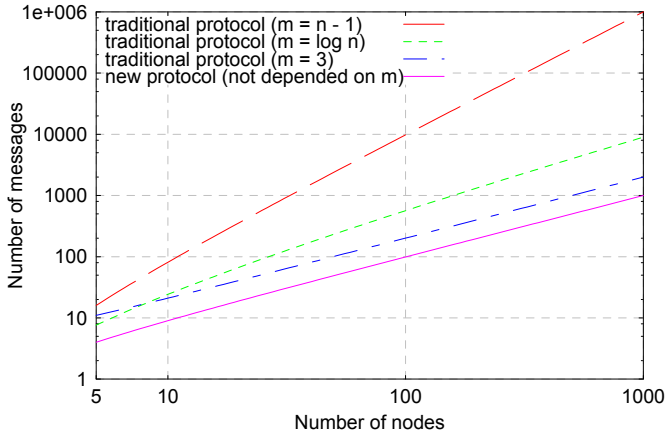


図 12 ノード数の変化によるメッセージ数

図 12は，ノード数の増加によるメッセージ数の変化を示す．図中，$m=n-1$ は，既存の Gnutella 等の P2P プロトコルにおいて，フルメッシュのトポロジーで発生するメッセージ数を示す．$m=\log n$ と $m=3$ は，それぞれ，Chord の場合，ネイバーが固定の場合を示す．

本プロトコルは，はじめの 1 回目のメッセージ数こそ，既存の Gnutella 等と変わらないが，次回以降のメッセージ数は，最も少ないことがわかる．$n=100$ の場合，本プロトコルは，$m=n-1$ の場合に比べ，99%，$m=\log n$ と $m=3$ の場合は，それぞれ，82.5%，50.7%のメッセージ数を削減できることがわかる．

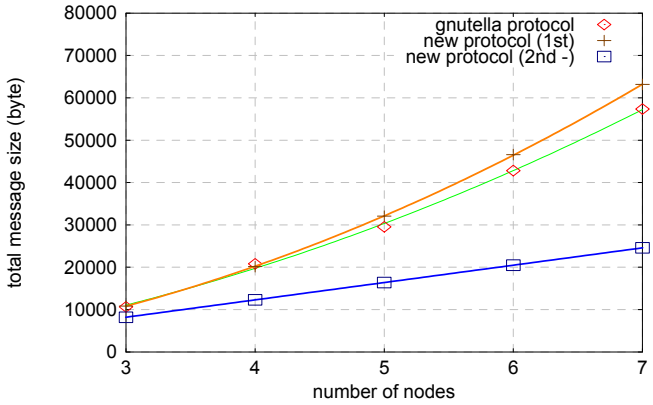


図 13　Gnutella とのメッセージ数比較

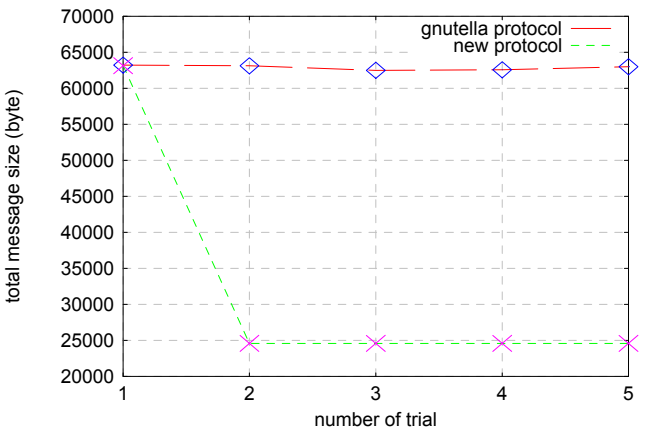


図 14　転送停止処理 2 によるメッセージ量の変化

6.2. プロトタイプによる評価

ノード数を 1 から 7 まで変化させた場合に，既存の Gnutella アプリケーションである Gnut と本プロトコルを実装したプロトタイプとのメッセージ量を比較した（図 13）．メッセージ量の測定は，各ノードで tcpdump を使用して全パケットを収集することで実現した．

Gnutella と本プロトコルの１回目のメッセージ量は，同程度であることがわかり，オーダは，共に $O(n^2)$であることがわかる．本プロトコルの２回目以降のメッセージ量におけるオーダは，$O(n)$であることを示している．

図 14は，要求送信回数ごとにどれだけのメッセージ量が発生するかを Gnutella と本プロトコルでそれぞれ測定した結果を示す．１回目のメッセージ量は，Gnutella と本プロトコルであまり変わらないが，本プロトコルは，２回目以降でメッセージ量が削減できていることがわかる．Gnutella は，２回目以降もメッセージ量が変化しない．

7. まとめと今後の課題

本稿では，既存の P2P プロトコルに比べ，トラヒック量を削減可能な BONCHI プロトコルを提案した．更に，本プロトコルのプロトタイプを作成し，既存の Gnutella アプリケーションと比較して，メッセージ数を削減できる有効性を示すことができた．

今後の課題としては，5章でネイバーを動的に選択する方法を提案したが，更に具体的な提案とプロトタイプでの評価により有効性を示すことを考えている．

8. 謝辞

本研究は，独立行政法人情報通信研究機構における委託研究テーマ「大規模ネットワークセキュリティの確保に向けた研究開発」によっている．ここに記して謝意を表す．

文　献

[1] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, “Protecting Free Expression Online with Freenet,” IEEE Internet Computing, vol. 6, pp.40-49, Jan.-Feb. 2002.

[2] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. “Freenet: A Distributed Anonymous Information Storage and Retrieval System,” Proc. Of the Workshop on Design Issues in Anonymity and Unobservability, pp.311-320, 2000.

[3] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, “Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service,“ Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), pp.71-76, May 2001.

[4] Y. Liu, Z. Zhuang, X. Li, and M. N. Lionel, “A Distributed Approach to Solving Overlay Mismatching Problem,” Proc. of the 24th IEEE International Conference on Distributed Computing System (ICDCS2004), pp.132-139, 2004

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,” Proc. of the 2001 ACM SIGCOMM Conf., pp.149-160, Aug.2001.

[6] M. Ripeanu. “Peer-to-Peer Architecture Case Study: Gnutella Network,” Proc. Of International Conf. on Peer-to-Peer Computing (P2P2001), pp.99-100, 2001

[7] “Freenet.” http://freenetproject.org/.

[8] “Gnutella.” http://gnutella.wego.com/.

[9] “Gnutellium.” http://www.gnutelliums.com/.

[10] “Gnut.” http://www.mrob.com/gnut/.

[11] 石田常竹，神尾政和，箱田貴久，塚本克治，清水弘，“トラフィック削減のための P2P サービスにおける経路制御手法，” 2004 電子情報通信学会総合大会，no.B-16-20, pp.629，Mar. 2004