情報処理学会論文誌 Vol. 52 No. 9 2709–2717 (Sep. 2011)

トラストを用いたハブノード強化手法 “Trusted-Hub”

松本愛咲[†1] 真下洋[†1]
安富正矩[†1] 重野寛[†2]

本論文では，オーバレイネットワークにおける転送拒否攻撃に着目し，悪意あるノードによる転送妨害の影響を抑止するためのハブノード強化手法を提案する．提案手法ではトラストを用いることで，悪意あるノードによる検知妨害を回避し，ハブノードの検知を正しく行いネットワークの冗長度を確保する．それとともに，検索クエリを送信するノードをトラストにより 1 つに決定し，検索効率を維持したままオーバヘッドを抑制する．また，ノード強化においてネットワーク密度を考慮することでノード強化を動的に行い冗長度を確保し，かつ経路構築にかかる無駄なメッセージを省きメッセージオーバヘッドを抑制する．提案手法の評価はシミュレーションにより行い，既存のハブノード強化手法に対しオーバヘッドの点で優位であることを示した．また，オーバヘッドを抑制した提案手法でも十分な検索成功率を得られた．

Trusted-Hub: Hub Node Reinforcement Method Using Nodes' Trust

Asaki Matsumoto,[†1] Yo Mashimo,[†1]
Masanori Yasutomi[†1] and Hiroshi Shigeno[†2]

In this thesis, Forwarding Obstruction Attack, an attack on overlay networks is focused on, and a hub node reinforcement method to deter affects of forwarding obstruction by malicious nodes is proposed. By using trust, each node deters obstruction of realizing as a hub node, and thus redundancy of network is ensured. Moreover, only one node forwarded search query is decided by using trust, and thus search success rate is kept and overhead is inhibited. Additionally, hub nodes guess density of network and decide adding node dynamically and inhibit wasteful messages to construct routes, and thus message overhead is inhibited. The results of computer simulation demonstrate inhibited overhead, small affect of obstruction of realizing as a hub node, and high success search rate.

1. はじめに

P2P 構造化オーバレイではネットワーク上に存在する全コンテンツを発見可能という利点がある[1)]．しかし構造化オーバレイに対する攻撃の一種に転送拒否攻撃がある．この攻撃は悪意あるノードが検索クエリの転送を怠るもので，コンテンツ要求の検索クエリを目的のノードに届かなくさせ，要求したコンテンツの発見を阻害するという問題がある．

構造化オーバレイにおける転送拒否攻撃への対策として，ルーティングテーブルを冗長化する手法がある[5)]．分散ハッシュテーブル（Distributed Hash Table，以下 DHT）[7),8)] における検索クエリ転送量の偏りに着目したハブノード強化手法[5)] もその 1 つである．ハブノードとは他ノードより多く検索クエリを転送するノードを指す．ハブノード強化手法では，各ノードが他ノードに転送回数の問合せを行い，自身の転送回数と比較して自身がハブノードかを判定する．ハブノードはすでにルーティングテーブルにあるノード間に宛先ノードを追加することでルーティングテーブルを拡張する．これにより，全ノードのルーティングテーブルを拡張するよりネットワーク負荷を抑え，かつ検索の成功率を維持できる．

しかし，既存のハブノード強化手法には 3 つの問題点がある．1 つ目は転送回数の偽装の問題である．転送回数の問合せに対し悪意あるノードが転送回数を偽り，本来ハブノードだと自己検知するべきノードが検知できず，ルーティングテーブルが拡張されないことから転送拒否攻撃への耐性が低下する．2 つ目は経路の冗長度不確保の問題である．すでに追加されているノードの ID 空間上の距離（以下，ネットワーク密度）を考慮せず追加するノードを静的に決めているため，同一ノードが追加される．同一ノードを追加することで冗長度は確保できず，かつ経路構築に掛かるメッセージオーバヘッドが増加する．3 つ目はネットワーク負荷の問題である．追加されたノードすべてに検索クエリを転送するため，経路強化したハブノードを経由する検索クエリ量は膨大になりネットワーク負荷が増大する．

問題解決のために，既存のハブノード強化手法では考慮されていなかった他ノードの転送回数や検索クエリを転送する経路の信頼性に焦点をあてる．大規模分散システムの P2P にはトラストがあり，一般にトラストとはノードの信頼性を評価した数値である[9),10)]．本論文では検索クエリの転送拒否攻撃を行うかどうかをトラストの基準とし，トラストを用いる

†1 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†2 慶應義塾大学大学理工学部
Faculty of Science and Technology, Keio University

ことで転送拒否攻撃を行うノードにクエリ転送することを回避できると考えられる．

そこで，本論文ではトラストを用いたハブノード強化手法 “Trusted-Hub” を提案する．これは，転送回数を偽造するような悪意あるノードのトラストは低いという仮定のもとに考えられた．Trusted-Hub では転送回数の偽装とネットワーク負荷の問題について，トラストを用いて解決する．すなわち，トラストにより転送回数偽装の影響を低下させ，トラストを考慮して転送先ノードを 1 つ選択する．さらに，トラストで転送拒否攻撃を行うノードを回避であるが，安全な経路確保のためにルーティングテーブルの強化が必要である．このとき，各ハブノードがネットワーク密度を推測し，ルーティングテーブルに追加するノードを動的に決定し，経路の冗長度不確保の問題を解決する．

以下本論文では，2 章において関連研究について述べる．3 章でトラストを用いたハブノード強化手法 Trusted-Hub を提案し，4 章でシミュレーション評価における Trusted-Hub の有用性を示す．最後に 5 章で結論を述べる．

2. 関 連 研 究

関連研究として，転送拒否攻撃，ハブノード強化手法，トラストについて説明する．

2.1 転送拒否攻撃

オーバレイネットワークにおける攻撃の一種に転送拒否攻撃が考えられている[5]．DHT[3),6)] におけるコンテンツ要求は，検索クエリがいくつかのノード間で転送されることにより行われており，転送途中で悪意あるノードにクエリが行き着いた場合，検索クエリの転送拒否や情報を改ざんした検索クエリの転送が可能になる．

一般に DHT を利用したネットワークにおいて，悪意あるノードの増加にともない検索クエリが正常に転送される確率が低下する[2]．また検索クエリの転送に要するホップ数もこの確率に影響を及ぼすため，スケーラビリティの観点から改善の必要がある．ここでクエリの成功を，検索クエリが目的のノードまでルーティングされ，クエリ発行ノードが目的のノードを認識できることと定義し，クエリ成功率を総クエリ数に対する成功クエリ数の割合と定義する．

2.2 ハブノード強化手法

ハブノード強化手法は，DHT ネットワークにおいてハブノードの存在に着目した転送拒否攻撃の影響を効率的に抑制する手法である．DHT ネットワークにおいて，検索クエリ転送のホップ数が多いほど転送拒否攻撃を行うノードを経由する確率が上がる[2]．したがってルーティングテーブルを拡張し，少ないホップ数で転送することが有効である．しかし，全ノードのルーティングテーブルを拡張するとネットワーク負荷が大きくなる．そこで検索クエリの転送回数が他ノードより多いハブノードに限定してルーティングテーブルを拡張することで，効率的に転送拒否攻撃を行うノードに検索クエリを転送する確率を低下させる．

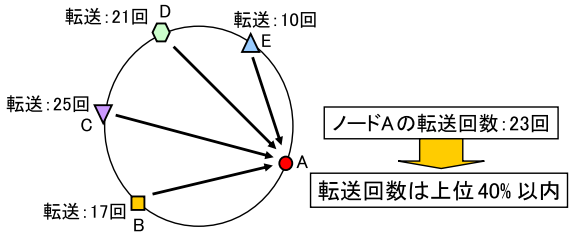

図 1　ハブノード自己検知の概略図
Fig. 1　Concept of Hub node self-detection.

この手法は各ノードが，自身がハブノードであるかを判断する自己検知処理と，ハブノードだと検知した場合のルーティングテーブル強化処理の 2 つの手順からなる．

2.2.1 ハブノードの自己検知

各ノードは，ルーティングテーブル更新時ハブノードの自己検知を行う．ハブノードの自己検知は，他ノードとの検索クエリ転送回数の比較により行われる．P2P において全ノードの転送回数を集約しソートするのは現実的でないため，ランダムに選択したノード間で局所的に転送回数の比較を行う．他ノードの検索クエリ転送回数は，問合せによって得る．

図 1 は，ノード A が転送回数に関する問合せを 4 つの他のノード（B〜E）に対して行った様子を表している．ノード A はこの 4 つの転送回数の情報から，自身の転送回数がネットワークの中で上位何%以内に位置づけられるか推定する．図 1 ではノード A よりも大きな転送回数を持つノードはノード C のみであり，ノード A は自身も含めた 5 つのノードの中で，上位 40%以内の転送回数を持つことが分かる．ハブノードだと検知される閾値より上位であれば，ノード A は自身をハブノードであると検知する．

2.2.2 ルーティングテーブルの強化

ハブノードのルーティングテーブルのみを強化（冗長化）することで，オーバヘッドに対して効率的にクエリ成功率を上昇させる．次式で決定される ID（$addID_{k_i}$）に対応するノードが新たな経路としてルーティングテーブルに追加される．

$$addID_{k_i} = ID_{start} + I_{base} \times i \quad (0 \leq i \leq \beta). \tag{1}$$

ID_{start} は元のルーティングテーブルの StartID，I_{base} は基本強化間隔と呼ばれる，パラ

メータとして与えられる適当な値である．また，β はルーティング強化倍率と呼ばれ，ルーティングテーブルを何倍に拡張するかを表す．ここで StartID とは，ルーティングテーブルに確保すべきノード ID の理論値を指す．必ずしも StartID を持つノードが存在しないため，StartID と等しいまたは大きい ID を持ち最も距離の近いオンラインノード（Successor）がルーティングテーブルに追加される．なお，ノードのルーティングテーブルへの追加は，その DHT アルゴリズムの通常のルーティングテーブル構築と同じ手順で行われる．

2.3　ハブノード強化手法の問題点

ハブノード強化手法には以下に示す 3 つの問題点がある．

- 転送回数偽装に対する耐性がない点
- 基本強化間隔 I_{base} に適切な値を設定しないと期待された冗長度を確保できない点
- クエリ量増加によってネットワークへの負荷が増大する点

2.3.1　転送回数偽装に対する耐性

ハブノード自己検知における悪意ある行動として，転送回数を実際の値より小さく申告する過小申告と，転送回数を実際の値より大きく申告する過大申告と，転送回数のいかんにかかわらず自身をハブノードと検知しない過小評価の 3 つが考えられる．

過小申告する行動については，問合せを行ったノードの転送相対度数の算出に影響を及ぼさない．したがって本来ハブノードと検知するべきノードの自己検知を妨げず，ネットワークの冗長度を低下させないため影響がないといえる．一方，過大申告した場合，問合せを行ったノードの転送相対度数が正しい値よりも小さくなり，ハブノードであるべきノードが自身をハブノードと検知できなくなる可能性がある．また，ハブノードの中には過小評価を行うノードがいることも考えられる．これはネットワークからルーティングテーブルを強化するノードが 1 つ減ることになり，ネットワークの冗長度が低下し転送拒否攻撃に対する耐性が弱まる．しかし，転送回数の過大申告が複数のノードの自己検知に影響を与えるのに対し，この場合には 1 つのノードの自己検知にのみ影響を与える．したがって，転送回数の過大申告よりクエリ成功率に与える影響は小さいと考えられる．

以上より，本論文では転送回数の過大申告を転送回数の偽装と定義する．なお，転送回数の偽装を行う動機はネットワークのクエリ成功率を低下させることにあり，転送拒否攻撃の動機と合致する．したがって，転送拒否攻撃を行うノードは転送回数の偽装も行う可能性が高いと考えられ，転送回数の偽装への対策を講じる必要がある．

2.3.2　基本強化間隔 I_{base} の設定

ハブノード強化手法では，ルーティングテーブルに基本強化間隔 I_{base} の整数倍にあたる ID に対するエントリを追加する．I_{base} がネットワーク密度に対し小さい値をとった場合，すでにルーティングテーブルに存在するノードを追加してしまう．エントリにすでにルーティングテーブルに存在するノードが追加されると，期待された冗長性を確保できないだけでなく，経路構築にかかる無駄なメッセージが生じる．したがって，すでにルーティングテーブルに存在するノードと同一のノードがエントリに追加されないように I_{base} を設定する必要がある．

しかし既存のハブノード強化手法の研究では，I_{base} をパラメータとして与え，その決定方法について議論していない．各ノードは ID 空間におけるノード配置の把握が非常に困難であるため，適切な I_{base} を設定できない．以上より，ネットワークの密度に応じて，適切な I_{base} を設定する仕組みが必要だといえる．

2.3.3　クエリ量増加によるネットワークへの負荷

ハブノード強化手法は，ルーティング強化倍率 β を用いてハブノードを通過するクエリを β 倍に増加させ，エントリ番号 k の ID，$addID_{k_i}$ すべてにクエリを転送し，いずれかの検索クエリが目的ノードに転送される．これによりクエリ成功率は上昇するが，各ノードのクエリ転送量が増加する．ハブノードは他ノードより多くのクエリを扱うノードであるため，クエリが増加する機会も多く，クエリ転送量の点から大きな問題となる．

2.4　トラスト

一般に，トラストは P2P ファイル共有における悪意ある行為の対策として用いられる．トラストはネットワーク上のノードの信頼性や安全性を評価し，その情報を複数のノードで共有することで悪意ある行為を行うノードとのファイル取引を回避するシステムである[4]．

各ノードはノードの信頼性や安全性を評価した結果であるトラストを数値として保持する．これを本論文ではトラスト値と呼ぶ．トラスト値はローカル値とグローバル値の 2 種類ある．ローカル値とは検索クエリ転送の成否により算出される値であり，クエリ発行者がクエリを転送したノードに対し算出する．グローバル値は各ノードが算出したローカル値をもとに算出される値であり，全ノードで共有される．本論文ではグローバル値をトラスト値と呼ぶ．なお，本論文では検索クエリの転送拒否攻撃を行うかどうかをトラストの基準とする．

3.　Trusted-Hub の提案

本論文では Chord において転送拒否攻撃の影響を抑止するために，トラスト値を用いたハブノード強化手法 “Trusted-Hub” を提案する．

転送回数偽装とネットワーク負荷の問題を解決するために，ノードの信頼性を評価したト

ラスト値を導入することで，転送先となるノードや各ノードが報告する転送回数が信頼できるかを判断する．すなわちトラストにより転送回数偽装の影響を低下させ，トラストを考慮して転送先ノードを 1 つ選択する．またトラストで転送拒否攻撃を行うノードを回避できるが，安全な経路確保のためにルーティングテーブルの強化が必要である．このとき，各ハブノードがネットワーク密度を推測し，ルーティングテーブルに追加するノードを動的に決定し，冗長度不確保の問題を解決する．

Trusted-Hub は，トラストの算出と以下の 3 つのフェーズからなる．

(1)　ハブノード自己検知フェーズ
(2)　ルーティングテーブル拡張フェーズ
(3)　経路選択フェーズ

フェーズ (1) で転送回数偽装の影響を抑制し，フェーズ (2) でルーティングテーブルの冗長性を確保し，フェーズ (3) でクエリ量低減を図る．なお，提案手法は Chord の適応を前提する．

3.1　トラスト値の算出

各ノードは，転送拒否攻撃を行うかどうかを基準に他ノードのトラスト値を決定しており，検索クエリ転送の成否をもとにローカル値を算出する．初期ローカル値を整数 0 とし，検索クエリ転送が成功すると，検索クエリが転送された経路上にあるノードのローカル値を 1 インクリメントする．検索クエリ転送が失敗すると，経路上にいるノードすべてが疑わしいとしてローカル値は変化させない[⋆1]．本論文では転送拒否攻撃を行うノードのローカル値は上がらず，他ノードと比べ低くなるため，ローカル値を集約したトラスト値もまた低くなる．トラスト値を算出する際，各ノードは自身が持つローカル値を以下の式を用いて正規化する．

$$S_{ij} = \frac{L_{ij}}{\sum_j L_{ij}}. \tag{2}$$

ここで，クエリ発行ノード i におけるクエリ転送ノード j に対するローカル値を L_{ij} とし，正規化したローカル値を S_{ij} とする．各ノードが保持する S_{ij} を集約し，トラスト値を算出する．なお，集約方法については本論文では議論しない．

⋆1 Chord には様々な実装が考えられるが，本論文ではクエリ送信ノードがクエリ転送経路を判別できるプロトコルを前提とする．

3.2　ハブノード自己検知フェーズ

このフェーズでは，各ノードが自身がハブノードであるかを判定する．このフェーズの目的は，ハブノードと検知するべきノードが，自身をハブノードと正しく検知することである．ランダムに選択したノード間で局所的に転送回数を比較するとき，自己申告される各ノードの転送回数の信頼性を考慮するためにトラスト値で重み付けをしている．これにより，トラスト値が低いとされる悪意あるノードによる転送回数の偽装に対処することができる．

以下に処理手順を示す．

(1)　転送回数の問合せ

各ノードはハッシュ関数を用いて，転送回数の問合せを行う宛先 ID を決定する．問合せを行う宛先 ID の数を n とすると，宛先 ID は次式で与えられる．なお，ID_{self} とは，転送回数の問合せを行うノード自身の ID を示す．

$$reqID_i = hash(ID_{self} + i) \quad (1 \le i \le n). \tag{3}$$

求められたすべての宛先 ID に対して，転送回数の問合せを行う．問合せを受けたノードは，自身の転送回数を返答する．問合せを行ったノードは，$reqID_i$ に対して返答したノード ID（$repNode_i$）とその転送回数 x_i を保存する．

(2)　転送相対度数の算出

転送回数の問合せが終了したノードは，得られた転送回数にトラスト値による重み付けを行う．重み付き転送回数 w_i は以下の式で求められる．

$$w_i = x_i \times trust(repNode_i). \tag{4}$$

ここで，$trust(repNode_i)$ は $repNode_i$ に対するトラスト値を示す．

次に，式 (5) より重み付き転送回数から転送相対度数を算出する．

$$R_{forward} = \frac{u}{n+1}. \tag{5}$$

ここで，u は重み付き転送回数が自身より多いノードの総数を表す．転送相対度数により，自身のネットワーク内での相対的な転送回数の多さを推測することができる．

(3)　ハブノードの検知判定

転送相対度数とハブノード判定閾値 α を比較して，自身がハブノードであるか判定する．すなわち，次式が成り立つとき，このノードは自身をハブノードと検知する．

$$R_{forward} \le \alpha \tag{6}$$

ここで α（$0 \le \alpha \le 1$）はハブノード判定閾値あり，重み付き転送回数が上位 α にあるノードをハブノードと判定させるためのパラメータである．

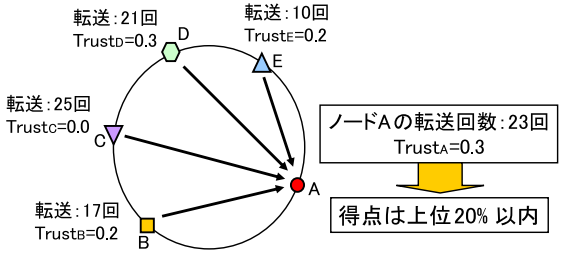

図 2　トラスト値を用いたハブノード自己検知
Fig. 2　Hub node self-detection using trust.

図 2 にノード A によるハブノード自己検知の例を示す．問合せの結果，他ノードの転送回数がそれぞれ {17, 25, 21, 10} 回，ノード A の転送回数が 23 回だと分かる．他ノードのトラスト値がそれぞれ {0.2, 0.0, 0.3, 0.2}，ノード A のトラスト値が 0.3 とすると，それぞれの重み付き転送回数 w_i は，式 (4) より {3.4, 0.0, 6.3, 2.0}，ノード A は 6.9 となる．式 (5) より転送相対度数は 0 となり，α=0.1 の場合，ノード A は自身をハブノードと検知する．たとえ悪意あるノードが転送回数を偽装しても，トラスト値の重み付けにより，ノード A は自身をハブノードと正常に検知できる．

3.3　ルーティングテーブル拡張フェーズ

このフェーズでは，ハブノードと検知したノードが自身のルーティングテーブルを拡張する．このフェーズでは，ネットワーク密度を推定し，すでにルーティングテーブルにあるノード間に等間隔にノードを新たに追加することで効率的に強化することを目的としている．

以下に処理手順を示す．

(1)　ノード間距離の推定

ハブノードは，自身のルーティングテーブルからネットワークにおけるノード間の距離を 2 のべき乗の形で推定する．すなわち，ノード間距離 $dist_{nodes}$ は次式で算出される．

$$dist_{nodes} = bitLength(\ distance(nextNode)\ ). \tag{7}$$

ここで，$bitLength(x)$ は数値 x のビット長を求める関数，$distance(nextNode)$ は直近の Successor との距離を表す．なお，Successor とはルーティングテーブルに確保すべきノード ID の理論値である StartID に相当するオンラインノードを指す．

ノード間距離 $dist_{nodes}$ は，ルーティングテーブルにエントリを追加する際に同一ノードを追加しないようにする目的で算出するため，隣接ノードとの距離をノード間距離として一般化してよいと考えられる．追加するエントリの ID 間隔を最低でも $2^{dist_{nodes}}$ 開けることにより，同一ノードが追加されることを避けることができる．

(2)　区間分割数の決定

ルーティングテーブルのエントリ番号 k に対して，区間分割数 d_k を決定する．ルーティングテーブルの各エントリのカバーする ID の範囲は，$[2^k, 2^{k+1})$ である．この範囲を d_k 等分し，追加する ID を決定する．区間分割数 d_k は次式で決定される．

$$d_k = \begin{cases} k - dist_{nodes} & (0 \le k - dist_{nodes} \le d_{max}) \\ d_{max} & (d_{max} < k - dist_{nodes}) \\ 1 & (\text{otherwise}) \end{cases} \tag{8}$$

ただし，d_{max} は区間分割数の最大値であり，パラメータとして与えられる．なお，ルーティングテーブルのエントリが隣接ノードである場合（$d_k = 1$）はそのエントリに対しては拡張せず，経路構築にかかる無駄なメッセージを減らすことができる．

(3)　エントリの追加

区間分割数 d_k から基本強化間隔 I_{base} を次式で算出する．

$$I_{base} = 2^{k-d_k}. \tag{9}$$

新たに追加されるエントリ番号 k の ID($addID_{k_i}$) は，以下の式で求められる．

$$addID_{k_i} = ID_{start} + I_{base} \times i. \tag{10}$$

ここで，ID_{start} は元のルーティングテーブルの StartID(2^k) を示している．

図 3 にルーティングテーブル拡張の例を示す．強化前の $k = 9$ に対するエントリに対し，強化後の表に示された 3 つのエントリが追加されている．このように強化前のエントリ間を満遍なく使うことで，同一エントリが追加される確率を低くしている．

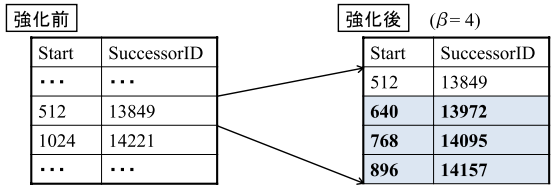

図 3　ルーティングテーブルの拡張
Fig. 3　Extension of Hub node’s rooting table.

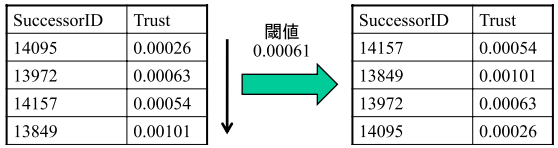

図 4　Cascade-Threshold による経路選択
Fig. 4　Route-selection by Cascade-Threshold.

3.4　経路選択フェーズ

このフェーズは，ハブノードがクエリを受信したときに実行される．追加された全ノードに検索クエリを転送すると検索クエリ量が膨大になりネットワーク負荷が増大するため，信頼できる経路を 1 つ選択し検索クエリを転送するために，このフェーズを実行する．信頼できる経路を選択するために，ノードのトラスト値を用いた選択アルゴリズムを使用する．

典型的なアルゴリズムとして，選択の候補となるエントリの中からランダムに選択し，そのノードのトラスト値が閾値より高ければ選択するという Random-Threshold が考えられる．しかし，このアルゴリズムではノードによって選択される回数に偏りがあると考えられるため，負荷分散の観点から Cascade-Threshold を考案した．Cascade-Threshold の流れは以下のとおりである．

(1)　選択の候補となるエントリを無作為に並べ替えておく．
(2)　先頭のエントリから順にトラスト値のチェックを行う．
(3)　先頭のエントリのトラスト値が閾値よりも高ければそのエントリを選択する．
(4)　選択が行われたら，チェックしたエントリを無作為に並べ替えて最後尾に挿入する．

このアルゴリズムの大きな特徴は，先頭のエントリから順にチェックしていき，最後に最後尾に挿入するところにある．これにより，1 度選択されたエントリは次回の経路選択フェーズで再選択される確率を低くでき，負荷分散を実現できると考えられる．また，トラスト値を閾値でチェックすることにより，トラスト値の低いノードを確実に回避することができる．なお本論文では，閾値の決定方法については議論しない．

図 4 に Cascade-Threshold による選択の例を示す．図 4 では，閾値 0.00061 に対してチェックを行っている．上から順にトラスト値のチェックを行うと，ID14095 は閾値より低く，ID13972 が閾値より高いことが分かる．したがって，ID13972 が選択され，ID14095 と ID13972 のエントリは無作為に並べ替えられて最後尾に挿入される．

3.5　ルーティングテーブル強化後の検索クエリの転送

ハブノードのルーティングテーブルを強化することで，転送経路が増加する．既存のハブノード強化手法では 1 つのエントリに対し増加した全経路に検索クエリを転送していたのに対し，Trusted-Hub では経路選択アルゴリズムに従い 1 つの経路を選択する．経路選択にトラスト値を用いることで転送拒否攻撃が行われる確率の高いノードを回避し，Chord と同量の検索クエリ数で高いクエリ成功率を実現させる．なお，ハブノードでない一般のノードの検索クエリ転送については，Chord，ハブノード強化手法と同様にスキップ検索を行う．

4.　シミュレーション評価

本章では，提案手法に関する各シミュレーション評価を示し，その考察を行う．

4.1　シミュレーション環境

シミュレーションに用いたパラメータを**表 1** に示す．本シミュレーションにおいて，悪意あるノードは検索クエリの転送を行わない（必ず転送拒否攻撃を行う）ノードを指す．また転送拒否攻撃を行うノードは，必ず転送回数の偽装を行うものとした．なお，本論文においてトラスト値は転送拒否攻撃を行うノードである確率が高いかどうかを示し，トラスト値が低いノードは転送拒否攻撃を行う確率は高いが，必ずしも行うわけではない．

シミュレーションではピリオドと呼ばれるタイムユニットが終了するごとにハブノードの自己検知フェーズを実行した．ここで検索の成功を，コンテンツ検索のために検索クエリを発行したノードが，目的のコンテンツを保持するノードを発見できることと再定義する．最大区間分割数は 3.3 節 (2) で述べた区間分割数の最大値である．最大区間分割数は，ハブノード強化手法と冗長度がおよそ等しくなるように設定した．

初めに経路選択アルゴリズムについて以下の項目で評価を行う．

- 選択したノードが転送拒否攻撃を行った回数の多さ
- 選択された回数の多さと分散

また以下の項目について，Chord，既存のハブノード強化手法，Trusted-Hub を比較する．なお，凡例の Chord，Hub，Trusted-Hub はそれぞれ，通常の Chord，既存のハブノード強化手法を実装した Hub Chord，提案手法を実装した Trusted-Hub Chord を表している．また，経路選択アルゴリズムは Cascade-Threshold を使用している．

- 転送拒否攻撃を行うノードの割合とクエリ成功率の関係
- 転送拒否攻撃を行うノードの割合とノードの平均クエリ転送回数の関係
- 転送回数偽装を行った場合のクエリ成功率の変化

表 1　シミュレーションパラメータ
Table 1　Simulation parameters.

パラメータ	値
Chord リングの ID 空間の大きさ　N	2^{32} (32 bit)
悪意あるノードの割合　p	0〜0.3
1 ピリオドあたりの検索クエリ発行回数	500
総ピリオド数	20
検索クエリの総発行回数	10,000
検索コンテンツの集中度	80%の確率で 20%のコンテンツを検索
ハブノード判定閾値　α	0.1
基本強化間隔　I_{base} (Hub)	2^{24}
ハブノード強化係数　β (Hub)	5
最大区間分割数　d_{max} (Trusted-Hub)	8

4.2　経路選択アルゴリズムに関する評価

あらかじめ 20 のエントリを用意し，転送拒否攻撃を行うノードの割合をパラメータで与え，1 回のシミュレーションで転送拒否攻撃を行うノードをランダムに割り当てる．全体でランダムに 10,000 クエリを発行し，目的ノードに検索クエリが到着したかの成否を求め，ローカル値を算出する．このローカル値を EigenTrust [4] によって集約し，算出したグローバル値をトラスト値として用いる．それらのノードの中から各アルゴリズムに従って 1 つのノードを選択する．この選択試行を 100 回繰り返し，選択されたノードが転送拒否攻撃を行った回数の割合と，各ノードが選択された回数の割合を算出し，各アルゴリズムを比較した．比較したアルゴリズムは，Random，Random-Threshold，Cascade-Threshold の 3 つである．Random は，すべてのノードの中から無作為に選んだノードを選択するアルゴリズムである．

4.2.1　選択されたノードが転送拒否攻撃を行った回数の多さ

前述の試行を行い，選択されたノードが転送拒否攻撃を行った回数の割合の平均値を算出した．なお，このシミュレーションにおいてのみ，各ノードが転送拒否攻撃を行う確率はトラスト値と負の比例関係にあるものとする．

表 2 に各アルゴリズムにおける転送拒否攻撃を行った回数の割合の平均値を示す．表 2 から，Random-Threshold，Cascade-Threshold ともに Random より転送拒否攻撃が行われる確率が低く，また，2 つの経路選択アルゴリズムには差があまりないことが分かる．この 2 つの経路選択アルゴリズムにおける結果に対し，有意の差がないと仮定して t 検定を行った．t 値は 0.64，自由度は 100 であり，有意水準 1%において両者に有意の差がないとした仮定は棄却できないことが分かった．

表 2　転送拒否攻撃を行った回数の割合
Table 2　Proportion of the number of Forwarding Obstruction Attack.

アルゴリズム	転送拒否攻撃を行った回数の割合
Random	0.315
Random-Threshold	0.174
Cascade-Threshold	0.176

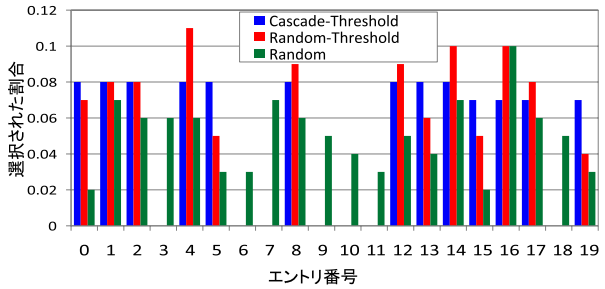

図 5　選択された回数の割合
Fig. 5　Proportion of the number of selection.

4.2.2　選択された回数の多さと分散

図 5 に各ノードの選択回数の割合を示す．図 5 から，Random が最も選択結果を分散できており，Random-Threshold と Cascade-Threshold では Cascade-Threshold の方が選択結果を分散できたことが分かる．Trusted-Hub の経路選択において，ノードの負荷分散の観点から選択結果が分散されることが望ましい．したがって，以上の比較評価から，Random-Threshold より Cascade-Threshold の方が Trusted-Hub の経路選択に適しているといえる．

4.3　転送拒否攻撃を行うノードの割合とクエリ成功率の関係

図 6 に転送回数の偽装がない場合の転送拒否攻撃を行うノードの割合とクエリ成功率の関係を示す．図 6 より，提案手法は通常の Chord よりもクエリ成功率について優位であることが分かる．転送拒否攻撃を行うノードの割合が 10%以上のとき，通常の Chord に比べてクエリ成功率が 5%程度上昇している．これは，フェーズ (2) による経路の冗長性の確保，およびフェーズ (3) におけるトラスト値を用いた経路選択の効果だと考えられる．

また，図 6 からは提案手法よりも Hub Chord の方がクエリ成功率について優れていること

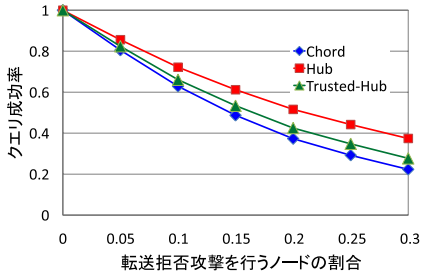

図 6　転送回数偽装がないときのクエリ成功率
Fig. 6　Query success rate without falsification of query transfer times.

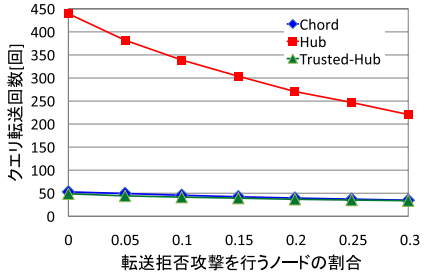

図 7　ノードの平均クエリ転送回数
Fig. 7　Average of query transfer times.

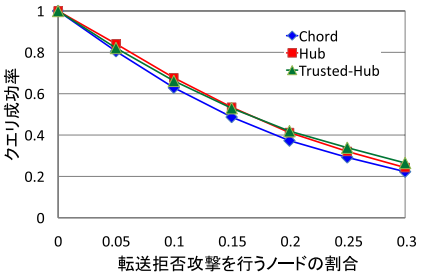

図 8　転送回数偽装があるときのクエリ成功率
Fig. 8　Query success rate with falsification of query transfer times.

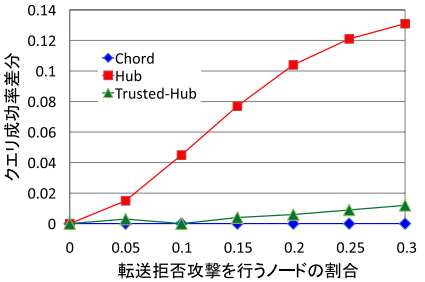

図 9　転送回数偽装の有無におけるクエリ成功率の差分
Fig. 9　The difference between the case with falsification of query transfer and without that.

が分かる．Trusted-Hub Chord のように 1 つの経路のみにクエリを転送する場合，ハブノードの次ホップに位置するノードによる転送は行われるが，さらにその次のノードが転送拒否攻撃を行うノードである可能性がある．したがって，1 つの経路のみを用いる Trusted-Hub Chord よりも，複数経路を用いる Hub Chord の方が高いクエリ成功率を維持できると考えられる．しかし Hub Chord は，転送回数偽装がないと仮定し，検索クエリをコピーし複数経路に同一クエリを転送し，そのうちのどれか 1 つのクエリが目的ノードに到着すれば成功としているため，現実的なシミュレーションシナリオで行う必要がある．

4.4　転送拒否攻撃を行うノードの割合とノードの平均クエリ転送回数の関係

図 7 に転送拒否攻撃を行うノードの割合とノードの平均クエリ転送回数の関係を示す．図 7 より，Trusted-Hub Chord は Hub Chord に比べて平均クエリ転送回数を抑制できたと分かる．最も効果が高いのは転送拒否攻撃を行うノードの割合が 0%のときで，Trusted-Hub Chord は Hub Chord のおよそ 1/9 まで平均クエリ転送回数を削減できている．ここで，転送拒否攻撃を行うノードの割合が高くなるにつれて全手法で平均クエリ転送回数が低減されているのは，転送拒否攻撃により転送されるべきクエリ量が減少することに起因する．

また，図 7 からは，Trusted-Hub Chord が通常の Chord よりも平均クエリ転送回数を低減できたといえる．これは Trusted-Hub のフェーズ (2) でのルーティングテーブル拡張とフェーズ (3) での経路選択に起因する．Trusted-Hub では，元のルーティングテーブルのエントリよりも大きい ID に対するエントリを追加している．そのため，それらのエントリが選択されたとき，目的ノードまでのホップ数が Chord よりも小さくなることが起こりうる．これにより，平均クエリ転送回数が減ったと考えられる．ホップ数の低下はわずかではあるが，転送拒否攻撃を行うノードの回避や検索にかかる時間にも効果的に作用すると考えられる．

4.5　転送回数偽装を行った場合のクエリ成功率の変化

図 8 に転送拒否攻撃を行うノードが転送回数偽装を行った場合の，転送拒否攻撃を行うノードの割合とクエリ成功率の関係を示す．図 8 から，転送拒否攻撃を行うノードの割合が 15%を超えるとき，Trusted-Hub Chord と Hub Chord のクエリ成功率が逆転していることが分かる．さらに，**図 9** に 4.3 節で評価した転送回数偽装のない場合とのクエリ成功率の差分を示す．図 9 から，Hub Chord が転送回数偽装の影響を大きく受けているのに対し，Trusted-Hub は転送回数偽装の影響をクエリ成功率にして約 1%以下に抑制できていることが分かる．以上から，転送回数偽装を考慮したとき，複数経路に同一クエリを転送する Hub Chord よりトラストを用いて単一経路を選択する Trusted-Hub の方が転送拒否攻撃に対して有効だと確認できた．

5. おわりに

本論文では，トラストとノード間距離を用いた Trusted-Hub を提案した．既存のハブノード強化手法の転送回数偽装とネットワーク負荷の問題を解決するためにトラストを導入し，転送回数や転送経路の信頼性を考慮した．さらに，各ハブノードがネットワーク密度を推測し，追加するノードを動的に決定することで，冗長度不確保の問題を解決した．

Trusted-Hub と既存のハブノード強化手法との比較評価を行った．経路選択方法について，Trusted-Hub には Cascade-Threshold が有効であることを確認した．Trusted-Hub は

最も効果が高いときで，Hub のおよそ 1/9 まで平均クエリ転送回数を削減できた．また，Trusted-Hub は Hub よりも転送回数の偽装に対する耐性があることが確認できた．以上より，Trusted-Hub はトラストを用いることで，既存のハブノード強化手法よりも転送回数の偽装に強く，検索のためのクエリ転送にかかる負荷を低減させた手法と結論付けられる．

今後の課題としては，転送拒否攻撃を行うノードと転送回数の偽装を行うノードを区別することがあげられる．複数のノードが結託し，役割を分担してこれらの攻撃を個別に行う可能性があると考えられる．そのため，悪意あるノードの行動モデルとして転送拒否攻撃と転送回数の偽装を区別して議論することが有効だといえる．

謝辞　本研究の一部はグローバル COE プログラム「アクセス空間支援基盤技術の高度国際連携」により行われました．

参 考 文 献

1) Balakrishnan, H., Frans Kaashoek, M., Karger, D., et al.: Looking up data in p2p systems, *Comm. ACM*, pp.43–48 (2003).
2) Castro, M., Druschel, P., Ganesh, A., et al.: Secure routing for structured peer-to-peer overlay networks, *OSDI2002*, pp.299–314 (2002).
3) Hildrum, K., Kubiatowicz, J., Rao, S., et al.: Distributed object location in a dynamic network, *Proc. 14th ACM SPAA 2002*, pp.41–52 (2002).
4) Kamber, S., Scholosser, M. and Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks, *ACM WWW 03*, pp.640–651 (2003).
5) Mashimo, Y., Shinzaki, Y., Ueda, S., et al.: Examination of forwarding obstruction attacks in structured overlay networks, *ARES 2008*, pp.1340–1345 (2008).
6) Maymounkov, P. and Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric, *Proc. IPTPS 2002*, pp.53–65 (2002).
7) Rowstron, A. and Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, *Middleware 2001*, pp.329–350 (2001).
8) Stoica, I., Morris, R., Karger, D., et al.: Chord: A scalable peer-to-peer lookup service for internet applications, *ACM SIGCOMM 2001*, pp.149–160 (2001).
9) Zhou, R. and Hwang, K.: Powertrust: A robust and scalable reputation sytem for trusted p2p, *IEEE TPDS 2007*, Vol.18, Issue:4, pp.460–473 (2007).
10) 松本愛咲，真下　洋，安富正矩，重野　寛：ID リストを用いた評価値集約手法 ILGT の提案，第 49 回 CSEC 研究発表会，Vol.2010-CSEC-49, No.13, pp.1–6 (2010).

(平成 22 年 11 月 30 日受付)
(平成 23 年 6 月 3 日採録)

松本　愛咲（学生会員）
2010 年慶應義塾大学理工学部情報工学科卒業．現在，同大学大学院理工学研究科博士前期課程在学中．P2P ネットワークの研究に従事．

真下　　洋（学生会員）
2008 年慶應義塾大学理工学部情報工学科卒業．2010 年同大学大学院理工学研究科博士前期課程修了．

安富　正矩（学生会員）
2009 年慶應義塾大学理工学部情報工学科卒業．2011 年同大学大学院理工学研究科博士前期課程修了．

重野　　寛（正会員）
1990 年慶應義塾大学理工学部計測工学科卒業．1997 年同大学大学院理工学研究科博士課程修了．1998 年同大学理工学部情報工学科助手（有期）．2003 年同大学理工学部情報工学科助教授．現在，同大学理工学部准教授．博士（工学）．情報処理学会学会誌編集委員，同論文誌編集委員，同マルチメディアと分散処理研究会幹事，同モバイルコンピューティングとワイアレス通信研究会運営委員等を歴任．ネットワーク・プロトコル，モバイルコンピューティング，ITS，ネットワーク・セキュリティ等の研究に従事．著書『コンピュータネットワーク』（オーム社），『ユビキタスコンピューティング』（オーム社）等．電子情報通信学会，IEEE，ACM 各会員．