Vol.2014-CSEC-65 No.30
Vol.2014-IOT-25 No.30
2014/5/23

Winnyネットワークにおけるクラスタリングを用いたインデックスポイゾニングの検討

油田 健太郎[1]　山場 久昭[2]　朴 美娘[3]　岡崎 直宣[2]

概要：現在，P2P ファイル共有ネットワークが世界中で利用されている．しかし，その多くはファイルの流通制御をもたないため，著作権侵害ファイルの流通やコンピュータウイルスによる個人情報の流出などが社会問題となっている．その解決策としてインデックスポイゾニングと呼ばれるファイル流通制御方式が研究されている．しかし，P2P ファイル共有ネットワークへインデックスポイゾニングを適用する際に，トラフィックの増大やインデックスの汚染などの問題が発生することが確認されている．そこで本論文では，ファイルの流通制御を低下させることなく，それらの問題を解決する手法として，動的なクラスタリングを提案することで従来手法を改善し，その一部を Winny ネットワーク向けに実装することで，提案手法の有効性を評価する．

キーワード：P2P，Winny，インデックスポイゾニング

1. はじめに

近年，通信技術の発達により常時接続可能で安価なブロードバンド回線が普及した結果，ブロードバンド時代に相応しい種々のコンテンツが普及した．P2P(Peer to Peer) 技術を用いたファイル共有ソフトウェアもそのひとつで，これは高速な回線を有効利用して効率的にファイルを送受信できることから注目されている．日本国内では，ネットワークを構成する各ノードがネットワークを維持するピュア P2P 型のネットワークを用いた，Winny や Share が広く利用されている [1][2]．現在，これらソフトウェアのネットワーク上では権利者の許諾を得ていない著作物の流通やコンピュータウィルスによる個人情報の流出が問題となっているが [3]，ネットワーク自体にファイルの流通制御を行う仕組みが備わっていないため，さらなる流通を食い止められないのが現状である．一方で，インターネットのトラフィック量は年々増加しており，日本国内のインターネットトラフィックは平成 24 年時点で 1.70Tbps に達している [4]．そして，その要因のひとつに前述したファイル共有ソフトが挙げられている [5]．特に，2010 年初頭に改正著作権法が施行された直後から国内の P2P トラフィックが激減していることが指摘されており [6]，著作物の違法な流通が P2P トラフィックを増加させている原因と推測される．以上の点から，トラフィック量を過剰に増加させない効率的なファイルの流通制御方式を検討する必要がある．

そこで，P2P ネットワークにおける流通制御方式として，ポイゾニングと呼ばれる手法が研究されている [7][8][9][10]．ポイゾニングには，本物のファイルに扮した偽のファイルを流通させるコンテンツポイゾニング，偽のファイル情報を拡散させるインデックスポイゾニング，ファイルのパラメータ部分を偽装するパラメータポイゾニングがある．なかでもインデックスポイゾニングは，流通制御に要するトラフィック量が比較的少ないため特に注目されている．そこで本研究では，仕様が明らかになっている Winny ネットワークにおいて，Winny 特有の仕組みであるクラスタリングを用いた重点的なインデックスポイゾニング手法を検討し，実際の Winny に対して実装して評価を行う．

2. Winny

2.1 ネットワークの概要

Winny [2] はピュア P2P 型のファイル共有ネットワークである．ピュア P2P ではノード同士の相互接続によりネットワークを構成する．ハイブリッド P2P のようにネットワークを管理する中央サーバを持たないため，耐障害性や拡張性に優れている．Winny ネットワークの各ノードは他ノードの情報やファイルのキャッシュ，ファイル情報を所

[1] 大分工業高等専門学校
Oita National College of Technology
[2] 宮崎大学
University of Miyazaki
[3] 神奈川工科大学
Kanagawa Institute of Technology

持している．このうち，ファイル情報をキーと呼び，キーには特定のファイルのファイル ID や位置情報 (ファイル所有者の IP アドレスとポート番号) などの各種メタデータが記録されており，キーを隣接するノード間で交換することでファイルの検索を行う．キーには寿命が設定されており，寿命が切れると次第にネットワーク上のインデックスから消去されていく．ファイルの所有者がネットワークに存在する間は新しいキーが継続的に拡散されるが，所有者がネットワークから離脱すると一定時間後にキーが消去されていく．つまり，あるファイルが所有者から取得できなくなれば，そのファイルのキーも自動的に削除される．また，Winny は他のピュア P2P ネットワークと同様にファイルの制御機構を持たないため，一度流通したファイルをネットワーク上から完全に削除することは極めて困難である．ここで，Winny ネットワークの概略を図 1 に示す．Winny のノード接続の種別には検索リンクと転送リンクがあり，前者ではファイル検索やキーの転送，後者ではキャッシュの転送を行うが，両者のプロトコルは共通のものである．検索リンクはネットワークの上流に対して最大 2 つ，自身と同等かそれ以下の下流に対して最大 5 つまで開くことができ，同時に 7 つのリンクが接続される．

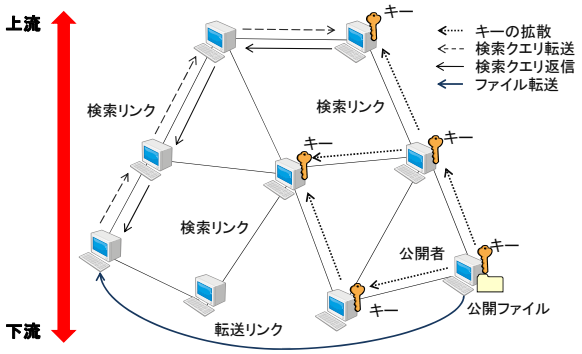

図 1　Winny ネットワークの概略

Winny には回線速度の速いノードが上流に，遅いノードが下流に位置する仕組みがあり，上流に行くほど頻繁かつ大量にキーやキャッシュのやり取りが行われる．したがって，高性能なノードがキーやキャッシュを多く保持することになり，多数の要求を受けても性能が低下しづらいため，効率的なファイル共有を実現している．なお，隣接ノードに申告する回線速度はユーザが任意の値を設定できるため，必ずしも実際の回線速度が反映されているとは限らない．

2.2　ファイル流通の仕組み

Winny では，ファイルをネットワーク上に公開するとき，ファイルからキャッシュと呼ばれるものを生成し，これと同時に，そのファイルのキーを生成する．公開されているファイルを取得するためには，そのファイルのキーが必要になるため，検索リンクを通してキーワードを設定した検索クエリを送信する．この検索クエリは 6 ホップ先のノードまで転送され，キーを持つノードはクエリにキーを付けて，同じ経路を通して検索元のノードにクエリを返送する．こうして得たキーの中にある位置情報をもとに，所有者に転送リンクで直接接続し，目的のファイルのキャッシュを得る．

これが基本的なファイルの流通のプロセスであるが，これには例外がある．Winny では，キーがノード間で転送されるときに，ファイルの位置情報が一定の確率で直前のノードのアドレスに書き換えられることがある．すると，そのノードは所有者でないのにキーの指すファイルのキャッシュ転送要求を受ける可能性がある．該当のファイルのキャッシュを持っていないノードは，オリジナルのキーを参照して，実際の所有者に転送リンクでキャッシュを要求する．そして取得したキャッシュを要求者に転送リンクで中継する．

このようにしてファイルの流通が進むと，あるファイルのキャッシュを複数のノードが所有することになり，ネットワーク上の該当ファイルのキーには複数の異なる位置情報が記録される．この状態になると，キーに記載されているファイル位置のノードが，そのファイルを初めにネットワーク上で共有したノード (一次放流者) であるのか判別できなくなるため，一次放流者の匿名性が確保される．

2.3　クラスタリング

一般的なピュア P2P ネットワークでは，嗜好性の異なるノードが混在しており，あるジャンルのファイルを検索するときに経路が複雑かつ冗長になるため，ファイルの検索効率が良いとは言えない．しかし Winny にはクラスタリングという仕組みがあり，接続するノードを選択するときに自分と嗜好の近いノードを優先的に選択する．このとき参照されるのが，各ノードが 3 つまで設定できるクラスタワードである．クラスタリングではクラスタワード同士の文字列的な距離を求めて，文字列の一致が多いほど評価値を大きくすることでノードの優先度を決定している．これによって，嗜好の近いノード同士が少ないホップ数で接続できるため，ファイル検索時の経路が各ノードに最適化されたものになる．このため，2.2 節で述べたように，Winny では検索クエリは 6 ホップ先のノードまでにしか転送されないが，クラスタリングの仕組みがあるため自分と嗜好の近いノードに対して十分に検索クエリを行き渡らせることができる．

2.4　プロトコルの概要

Winny が使用しているプロトコルについて説明する．ここで説明するプロトコルは，リンク種別によらず共通のものである．まず，パケットの構造を図 2 に，Winny プロ

トコルで定義されているコマンドの一部を表 1 に示す．パケットは通信ブロックを連結したもので，通信ブロックは先頭 4 バイトがブロック長，5 バイト目がコマンド番号，残りがコマンド引数で構成される．さらに，通信の機密性を確保するためにブロック全体が 32bit の RC4 で暗号化される．

ここで，Winny プロトコルにおけるリンクの確立プロセスについて説明する．あるノードと最初の通信を行うとき，先頭 2 バイトにダミーのバイト列，続けて 4 バイトに RC4 暗号の共通鍵を付加し，コマンド 00 からコマンド 03 を 1 つのパケットにまとめて送る．相手ノードがそのパケットを受け取ると，先頭に付けられている RC4 共通鍵でブロック全体を復号し，コマンドの内容が正しければ同じ手順で RC4 共通鍵とコマンド 00 からコマンド 03 を返信する．コマンド 00 の受信後に相手から受け取った共通鍵を変形し，以後の通信は同じ鍵で暗号化および復号化を行う．

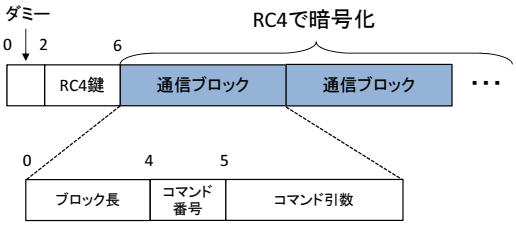

図 2　Winny プロトコルのパケットの構造

3. インデックスポイゾニング

3.1 インデックスポイゾニングの仕組み

インデックスポイゾニングとは，ネットワーク上に存在するファイルの情報を改変することで，そのファイルを取得できなくするファイル流通制御手法である．Winny では，本物のキーから表 2 のダミーキーを生成してそれを拡散することでインデックスポイゾニングを実現できる．ダミーキーには本物のキーと同じファイル名とファイルサイズが記録されているが，一方でファイル ID やファイルの位置情報は架空のものである．このダミーキーを受け取ったノードは，ファイルのダウンロード時に存在しない IP アドレスに対してキャッシュを要求するため，必ずダウンロードが失敗する．

インデックスポイゾニングは多くのノードに対して制御を行うことでより効果的に流通を制御することができるが，制御対象を拡大することでトラフィックの増加を招いたり，ネットワーク全体のインデックスが汚染されることがある．

表 1　Winny プロトコルのコマンド例

コマンド番号	内容
00	バージョン情報申告
01	回線速度申告
02	リンク種別申告
03	自ノード情報
04	他ノード情報
13	クエリ転送
21	キャッシュブロック転送
31	切断要求

表 2　ダミーキーの例

キー情報	内容
ファイル名	本物のキーと同様
ファイルサイズ	本物のキーと同様
ファイル ID	ランダム
ファイル本体の位置情報	存在しない IP アドレス

3.2 インデックスポイゾニングの問題点

インデックスポイゾニングでは，違法にアップロードされた著作物など，特定のファイルの流通を制御できるが，Winny を利用する正規のユーザに対して本来の Winny ネットワークの利用を制限するような影響を与えないことを目的としている．しかし現在，インデックスポイゾニングを P2P ネットワークに適用した場合にいくつかの問題点が報告されている．

基本的なインデックスポイゾニングでは，ファイルの位置情報を偽装したダミーキーをネットワーク上に大量に拡散することで，ネットワーク上の全てのノードへダミーキーを保持させる．これにより，特に多くのファイルに対して制御を行う場合には膨大な量のダミーキーを拡散させるため，制御に要するトラフィックが増大するという問題が発生する．また，拡散した大量のダミーキーがネットワーク上に長時間残ると，ネットワークのインデックスの大部分をダミーキーが占めることになり，それ以外のファイルが利用できるインデックスの領域が圧迫される．加えて，インデックスポイゾニングでは，本物のファイルと同じファイル名のダミーキーを拡散するため，正規のユーザが，制御対象ファイルと同じキーワードを持つ違法でないファイルを検索する際にまで，ダミーキーがヒットしてしまう．そのため，ファイル要求者である正規のユーザは目的のファイルを入手するために何度も検索を繰り返す必要がある．このように，大量のダミーキーが制御対象ファイル以外のファイルの流通を阻害する原因となっている．そのうえ，インデックスポイゾニングによる流通制御により DDos 攻撃を誘発することも指摘されている [11]．

3.3 関連手法

3.3.1 ファイル ID 検索に対するポイゾニング

ここでは，インデックスポイゾニングをファイル ID 検

表 3　ファイル ID 検索に対応したダミーキー

キー情報	内容
ファイル名	本物のキーと同様
ファイルサイズ	本物のキーと同様
ファイル ID	本物のキーと同様
ファイル本体の位置情報	存在しない IP アドレス

表 4　提案手法のダミーキー

キー情報	内容
ファイル名	本物のキーと同様
ファイルサイズ	本物のキーと同様
ファイル ID	本物のキーと同様
ファイル本体の位置情報	存在しない IP アドレス
生存時間	3600 [sec]

索に対応させた手法について説明する [8]．基本的なインデックスポイゾニングでは，制御対象ファイルと同じファイル名をもったダミーキーを拡散する．しかし，Winny ではポイゾニングの対策として，ファイルのファイル ID をキーワードとした検索が可能となっている．このファイル ID はファイルの内容から算出した MD5 ハッシュ値が使用されているため，ネットワーク全体で一意である．そのため，ファイル ID での検索に対して，ファイル名を同じにしたダミーキーでは本物のキーの入手を防ぐことができない．そこで，この手法では表 3 のようにダミーキーのファイル ID を制御対象ファイルと同じものにすることで，ファイル ID 検索に対してもインデックスポイゾニングを適用可能にしている．

ファイル ID 検索に対するポイゾニングでは，制御対象ファイルをアップロードするノードが持つキーをダミーキーで上書きすることにより，アップロードを行うノードから拡散する制御対象ファイルのキーはダミーキーとなる．これにより，最初に拡散したダミーキーがネットワーク上から消失した後でも，インデックスポイゾニングの効果を持続することができる．そのうえ，ファイル ID が異なるダミーキーを大量に拡散する必要がないため，この手法ではネットワークのインデックスが過剰に汚染されない．また，ダミーキーのファイルサイズの項目を，実際のファイルサイズより非常に小さい値にすることで，アップロードを行うノードがその値以上のアップロードを不可能にすることができる．

3.3.2　キーの生存時間を利用したインデックスポイゾニング

3.2 節で述べた Winny ネットワークにインデックスポイゾニングを適用する際に発生する問題を，キーの生存時間を利用することで解決する手法が提案されている [7]．キーには生存時間が設定されているため，ファイル保持ピアがネットワークから離脱すると自動的にそのキーもネットワーク上から消えていくことになる．この生存時間の規定値は 1500 秒前後であるが，任意に設定することもできる．また，Winny ネットワークでは既に保持しているキーと同じファイル ID を持つキーを受け取った場合に，それまで保持していたキーに上書きされるという特徴を持つ．これらの機能を用いて，Winny ネットワーク上の制御対象ファイルのキーをダミーキーで上書きし，キーの生存時間を 0 秒に設定することで，ネットワーク上に長時間ダミーキーを残すことなくインデックスポイゾニングを適用する方法である．拡散したダミーキーは，即座にネットワーク上から削除されるため，ダミーキーがネットワーク上で起こす問題の発生を防ぐことができる．

しかし，制御対象ファイルを保持しているピアが Winny アプリケーションを再起動すると再びキーの拡散が開始されるため，継続してダミーキーの拡散を行わなければならない．そのため，通常のインデックスポイゾニングよりも制御トラフィックが増加する可能性がある．

3.4　提案手法

本研究では，昨年度に提案された [12]，Winny に特化したインデックスポイゾニング手法を部分的に実装する．この提案手法では，表 4 のダミーキーを用いてポイゾニングを行う．従来手法 (表 3) と比較すると，この手法ではダミーキーをより長くネットワークに留まらせるために，生存時間を規定値より長めに設定している．さらに，制御ノードが所属クラスタを動的に変更することでポイゾニングの有効範囲を限定して効率的に流通制御を行い，制御に要するトラフィック量を削減する．

4.　提案手法の実装システム

4.1　概要

提案手法の一部を実装したシステムについてその概要を説明する．本研究では，システムを短期間で開発するために，Winny の公式クライアント (Winny v2.0b7.1) に対してメモリ書き換えやコードの注入などを行い，その動作を改変することでインデックスポイゾニングを実現した．そのためには，Winny の仮想アドレス空間に用意したコードを展開する必要がある．これを実現するために，システムを DLL(動的リンクライブラリ) として実装し，実行中の Winny にその DLL をロードさせる．この DLL を以後フック DLL と呼ぶ．フック DLL は，図 3 のように Winny の仮想アドレス空間に入り込み，Winny に対して以下の操作を行う．

（1）　Winsock API のフック

Windows の一般的なネットワークアプリケーションでは，TCP/IP 通信を行うのに Winsock という標準ライブラリを用いる．これは Winny も例外ではない．そして Winsock には，パケットの送信に

send()，パケットの受信に recv()，そしてソケットを閉じる closesocket() という関数が用意されている．本システムでは，Winny が送受信するパケットを傍受あるいは改変するために，Winny がこれらの関数を呼び出す際に割り込みを行い，フック DLL に組み込まれているコードを実行させる．

（2） 改竄チェックの無力化

Winny には，自身の実行ファイルとメモリに展開された自身のプログラムのシグネチャを計算し，改変を検出すると一定時間接続している隣接ノードを切断する機能が搭載されており，容易にプログラムの改竄が行えないようになっている．フック DLL はメモリ上の Winny のプログラムを一部書き換えるため，この機能の影響を受ける．そのため，フック DLL は自身が Winny にロードされたとき，改竄チェックを行っているプログラムのバイトコードを変更し，隣接ノードの切断が行われないようにする．

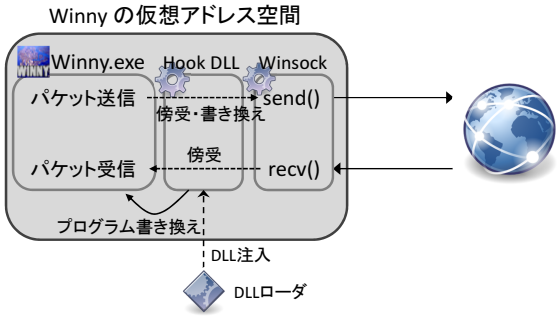

図 3　フック DLL の動作の概略

4.2　実装システムの機能

フック DLL は，Winny が 4.1 節 (1) で挙げた Winsock 関数を実行しようとしたときに，プロトコル解析やパケットの書き換えなどを行い，インデックスポイゾニングを実現する．ここでは，フック DLL に実装されている各機能について説明する．

（1） プロトコル解析

自ノードが送受信する Winny のパケットを復号化して，パケットに含まれるブロックやコマンドを認識し，その内容を解析する．フック DLL は，以下に述べる機能を実現するのに必要十分な Winny プロトコル解析機能を有する．

（2） ノード情報の収集

隣接ノードから Winny プロトコルのコマンド 03 または 04 を受け取った時に，コマンドに含まれるノードの情報を収集する．ここで得る情報は，ノードの IP アドレス，転送リンクのポート番号，申告速度，クラスタワードである．特に，クラスタワード情報は後述するクラスタワードの動的な変更に用いる．

（3） キーの収集

隣接ノードから Winny プロトコルのコマンド 13 を受け取った時にクエリを解析し，キーが含まれるクエリであればキー情報を収集する．

（4） キーのポイゾニング

自ノードが隣接ノードに制御対象のキーを含むクエリを送信しようとしたときに，制御対象キーに含まれる位置情報と生存時間を書き換えた新たなクエリを生成し，それを自ノードの RC4 共通鍵で正しく暗号化したパケットを送信する．

（5） クラスタワードの動的な変更

フック DLL は，30 秒毎に初期クラスタワードと最も頻繁に設定されているクラスタワードを設定し，Winny に反映させる．ここで比較対象になるクラスタワードはノード情報の収集で得たクラスタワードで，初期クラスタワードを c，あるノード i のクラスタワードの組を $x_{ij}(i = 1, \ldots, N, j = 1, 2, 3)$，収集したクラスタワードを $y_k(k = 1, \ldots, M)$ とすると，初期クラスタワードとの相関度 r_{y_k} の評価は (1)(2) 式で行う．ただし，y_k は c を含まない．

$$r_{y_k} = \sum_{i=1}^{N} f(c, x_{ij}, y_k) \tag{1}$$

$$f(c, x_{ij}, y_k) = \begin{cases} 1 & (x_{i1}, x_{i2}, x_{i3}\text{の} \\ & \text{いずれかが } c \text{ と等しく，} \\ & \text{かつ } y_k\text{が } c \text{ と共に} \\ & \text{設定されている)} \\ 0 & (otherwise) \end{cases} \tag{2}$$

4.3　ポイゾニングのプロセス

本研究の実装手法におけるポイゾニングのプロセスを説明する．

（1） 制御ノードを，制御対象ファイルに最も関連するキーワードを初期クラスタワードに設定した上でネットワークに参加させる．

（2） 制御ノードは，Winny ネットワークに参加しているノードの情報を収集しつつ，初期クラスタワードと

相関の高いクラスタワードを設定し，制御対象ファイルのキーが多く流通しているクラスタに参加する．その後，隣接ノードからキーを収集する．

（3） 制御ノードは，自ノードから隣接ノードに制御対象ファイルのキーを含むクエリを送信するときに，インデックスポイゾニングを行う． □

4.4 問題点

この実装手法には，キー改変の自由度が低いという問題点がある．これは Winny がパケットの暗号化に RC4 を採用しているためである．RC4 はストリーム暗号であり，暗号化に用いる疑似乱数配列の状態は 1 バイトの入力を行うごとに変化する．そして，この手法でポイゾニングを行うためには，フック DLL が Winny 内部と同じ状態の疑似乱数配列を持っていなければならない．すなわち，フック DLL が Winny から受け取るパケットと，フック DLL が送り出すパケットの長さは 1 バイトも異なってはならないということである．そのため，文献 [12] で提案されている，ファイルの人気度に応じてダミーキーの拡散量を動的に変更する手法は実現できない．

また，この実装手法では実装システムが制御対象ファイルの流通に加担する可能性がある．実装システムが組み込まれた Winny クライアントは本物のキーを持っており，キーの拡散の過程でキーの位置情報を自身のものに書き換えると，自ノードにキャッシュ転送要求が来ることになり，本物のキーに記載されたファイルの位置情報をもとに，キャッシュブロック中継動作を行ってしまうためである．現在の実装システムでは，Winny クライアントのキャッシュブロック中継動作を抑制できていない．

5. 評価

5.1 環境

本研究では，流通制御システムを実装した Winny クライアント (制御ノード) を VirtualBox 仮想マシンで構成された小規模な Winny ネットワークに参加させ，測定ノードからすべての制御対象ファイルのダウンロードを試行する際に要する時間を比較することで，提案手法の有効性を評価する．評価環境の概略図を図 4 に示す．

Winny ネットワークを構成するノードは 23 ノードで，一般ノードは 3 種類のクラスタに編成されており，それに加えていずれのクラスタにも未所属の中流ノードが 5 台存在する．各クラスタは上流ノード 2，中流ノード 2，下流ノード 1 で構成される．一般ノードはそれぞれ 20〜21 個の異なる制御対象外ファイルと，28〜29 個の制御対象外の著作物ファイルを所有しており，ネットワーク上には 572 個の制御対象外ファイルと 408 個の制御対象外の著作物ファイルが存在する．一方，制御対象ファイルを所有するノードは 20 個の制御対象の著作物ファイルを送信可能な状態にする．よって，ネットワーク上に存在するファイルの総数は 1000 個である．ここで，制御対象外ファイルと著作物ファイルの比率は，文献 [3] を参考にし，著作物ファイルが全体のおよそ 42.75% を占めるように設定し，制御対象の著作物ファイルの名前は “CopyrightedA[1-20 の連番]”，制御対象外の著作物ファイルの名前は “CopyrightedB[1-408 の連番]”，制御対象外ファイルの名前は “Public[1-572 の連番]” とする．制御ノードはネットワークに上流ノードとして参加させ，制御対象ファイルに対するインデックスポイゾニングを実行する．また，測定ノードはネットワークに中流ノードとして参加させ，制御対象ファイルのダウンロード試行を行う．

これに加えて，各クラスタの中流ノードの一つを観測点としてダミーキーの総数を測定することで，各クラスタのインデックスの汚染度を評価する．インデックスの汚染度が高いほど，測定ノードがダミーキーを取得する確率が高まるため，ダウンロード所要時間が増加する．ここで，インデックスの汚染度はそのノードが取得した全ファイルのキー A に含まれる制御対象ファイルのダミーキー B の割合で算出する．本研究の評価環境では，理想的なインデックス汚染度 p_{id}[%] は次式から求まる．ただし，実際には A が 1000 に満たないため，場合により p_{id} はこの値を超えることがある．

$$p_{id} = \frac{B}{A} * 100 = \frac{20}{1000} * 100 = 2.0[\%] \qquad (3)$$

なお，制御ノードは提案手法に従ってクラスタワードを動的に変更することがあり，ネットワークに流通するファイルのサイズはすべて 1MB である．また，上流ノードの申告速度は 300KB/s，中流ノードの申告速度は 120KB/s，下流ノードの申告速度は 50KB/s に設定する．加えて，ネットワーク上の全ノードで，Winny.ini の AutoDisconnect を 1 に設定することで，検索リンクの自動切断頻度を上げ，実際の Winny ネットワークに近い状態を再現する．

5.2 シミュレーション条件

シミュレーションに用いる条件について説明する．本研究では，シミュレーション条件として以下の 4 つの条件を設定し，それぞれの条件で 5 回シミュレーションを行う．ここで，条件 (3) と (4) は，制御対象ノードが属するクラスタの範囲を変化させることで発生する制御効率の相違を検証するために設定した条件である．

（1） 制御対象ノードは初期クラスタワードに “Target” を設定し，制御ノードはポイゾニングを行わない．測定ノードは，制御対象ファイルのキーがネットワーク上に十分に拡散されてからダウンロード試行を開始する．

Vol.2014-CSEC-65 No.30
Vol.2014-IOT-25 No.30
2014/5/23

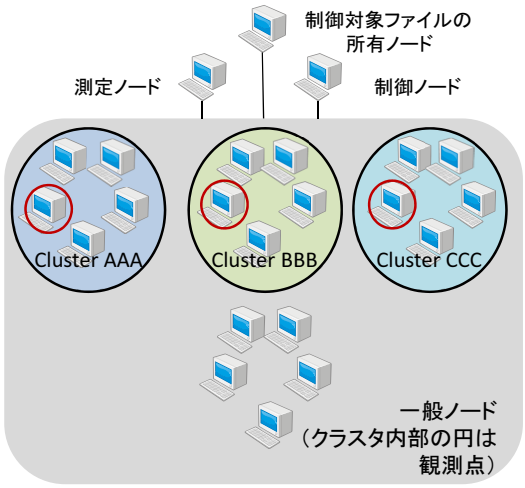

図 4　評価環境の概略図

（2）　制御対象ノードはネットワーク上のクラスタに参加せず，制御ノードは初期クラスタワードに “Target” を設定して，動的にクラスタワードを設定しない従来手法によるポイゾニングを行う．測定ノードは，制御対象ファイルのキーが十分にポイゾニングされてからダウンロード試行を開始する．

（3）　制御対象ノードは初期クラスタワードに “Target” と “BBB” を設定し，制御ノードは初期クラスタワードに “Target” を設定して，提案手法によるポイゾニングを行う．測定ノードは，制御対象ファイルのキーが十分にポイゾニングされてからダウンロード試行を開始する．

（4）　制御対象ノードは初期クラスタワードに “Target”，“BBB” と “CCC” を設定し，制御ノードは初期クラスタワードに “Target” を設定して，提案手法によるポイゾニングを行う．測定ノードは，制御対象ファイルのキーが十分にポイゾニングされてからダウンロード試行を開始する．

ここで，ダウンロード試行は，測定ノード上で名前に “CopyrightedA” を含むファイルをダウンロードする条件を追加して自動で行う．なお，シミュレーション時間は最大 1 時間とし，1 時間を超過するとダウンロード試行に失敗したものとする．また，各シミュレーションの終了後に必ず全ノードを終了させ，キャッシュフォルダの内容を消去することで，各ノードが行うキャッシュブロック中継動作が次回のシミュレーションに影響しないようにする．

5.3　評価結果

全制御対象ファイルのダウンロード試行の所要時間の平均を図 5 に示す．図 5 中のバーはダウンロード所要時間の最大値と最小値を示す．このグラフにおいて，ポイゾニングを行った場合 (2)(3)(4) のダウンロード所要時間はポイゾニングを行わない場合 (1) の所要時間と大幅に差がついており，インデックスポイゾニングを用いたダウンロードの阻害には成功しているものと考えられる．次に，従来手法 (2) と提案手法 (3)(4) の結果を比較すると，提案手法は従来手法より平均で 58% ほどダウンロード所要時間を増加させることに成功していることがわかる．また，提案手法 (3) と (4) を比較すると，提案手法 (4) はダウンロード所要時間の最大値が最も高いものの，平均値には有意な差が見られなかった．

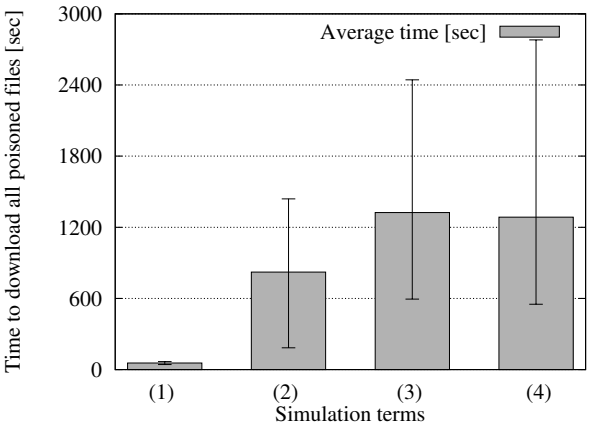

図 5　全制御対象ファイルのダウンロードに要した時間の平均

次に，従来手法 (2) と提案手法 (3)(4) のシミュレーション結果から最も平均値に近い時の結果 (表 5) のみを取り上げて，その結果が得られた状況において各クラスタの観測点で測定したダミーキーの総数を基に，各クラスタのインデックス汚染率を評価した．従来手法 (2) におけるインデックス汚染率を図 6，提案手法 (3) におけるインデックス汚染率を図 7，提案手法 (4) におけるインデックス汚染率を図 8 に示す．なお，これらのグラフの始点は観測点でダミーキーが観測される直前であり，ダウンロード試行を開始する直前ではない．提案手法では，初期段階から制御対象ノードが所属するクラスタに対して重点的なポイゾニングが行われていることがわかる．また，提案手法と従来手法を比較すると，従来手法はインデックス汚染率の推移が不安定であることがわかる．

6.　まとめ

6.1　考察と今後の課題

本研究では，P2P ファイル共有ネットワークにおけるファイル流通制御方式であるインデックスポイゾニングにおいて，制御効率を低下させることなく制御トラフィック

表 5　各シミュレーション条件におけるダウンロード所要時間の代表値

シミュレーション条件	所要時間 [sec]
制御なし (1)	57
従来手法 (2)	947
提案手法 (3)	1073
提案手法 (4)	1325

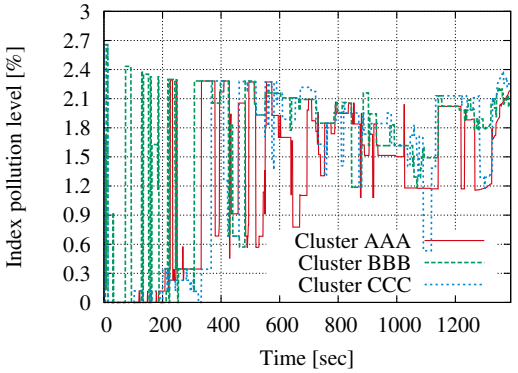

図 6　従来手法 (2) におけるインデックス汚染率

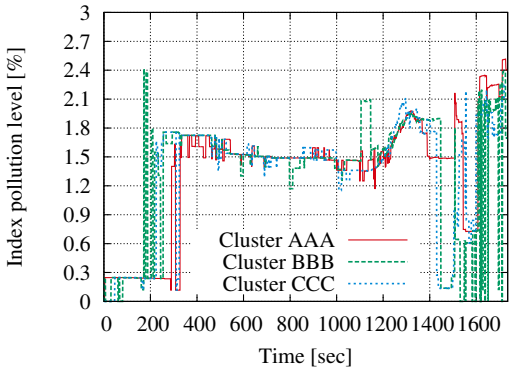

図 7　提案手法 (3) におけるインデックス汚染率

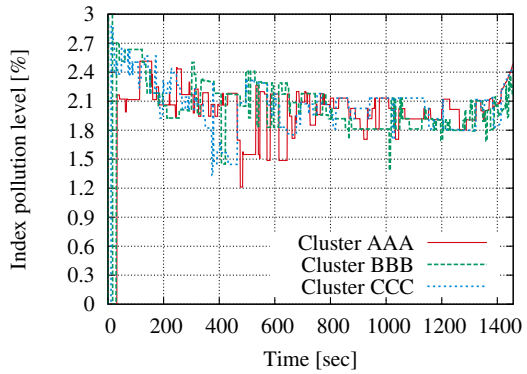

図 8　提案手法 (4) におけるインデックス汚染率

量を削減するために，クラスタリングを用いた重点的なポイゾニング手法を提案し，実際に Winny ネットワークに実装することで提案手法の評価を行った．

シミュレーションの結果，図 5 より提案手法は動的にクラスタワードを設定しない従来手法と比較して平均で 58% ダウンロード所要時間を増加させることができた．提案手法では制御対象ノードが参加しているクラスタに動的に参加することで，重点的なポイゾニングを行えるため，結果として制御効率が向上したものと考えられる．このことは，提案手法を実行した場合のインデックス汚染度と，従来手法の場合のインデックス汚染度を比較しても明らかである．従来手法 (図 6) におけるインデックス汚染度は 1.2% から 2.1% までを不安定に推移しているが，提案手法 (図 7，図 8) では安定して 1.5% 以上を維持できている．また，図 7 と図 8 を比較すると，提案手法 (4) が最もインデックス汚染度の平均値が高いことがわかる．ここで，シミュレーション条件 (3) と (4) の違いは制御対象ノードが参加するクラスタの範囲であり，(3) では “BBB”，(4) では “BBB” と “CCC” である．Winny のクラスタリングではクラスタワード間の文字列の一致が多いほど相関度が高くなるため，制御対象ノードが設定しているクラスタワードが複雑であればあるほど制御ノードと制御対象ノード間の関連性が高くなり，提案手法の制御効果を向上させると考えられる．なお，図 7 と図 8 では，シミュレーション条件 (3) と (4) においてクラスタ “BBB” や “CCC” に対する重点的なポイゾニングを行っているにもかかわらず，時間経過によりクラスタ “AAA” でも他のクラスタと同等のインデックス汚染率が得られているが，これは各クラスタの上流ノード間でも定期的にキーの交換が行われるため，他のクラスタに存在するダミーキーが次第にクラスタ “AAA” にも反映されたものと考えられる．

一般に，インデックスポイゾニングの実装システムでは複数の制御ノードをネットワークに参加させることで流通制御を行う [7][8][9]．そのため，制御ノードの台数だけ制御トラフィックが増大するが，本研究の提案手法を用いることで制御ノードあたりの制御効率が向上するため，制御ノードの台数を減らすことができ，制御トラフィック量の削減が期待できる．しかし，実際の Winny ネットワークは評価環境よりも動的なネットワークであり，実際は多種多様なファイルの共有が行われているうえ，ノードの参加や離脱が頻繁に繰り返されることで，クラスタワード間の相関の変化が起こる．特に，評価環境は実際の Winny ネットワークよりも小規模であり，提案手法により動的にクラスタワードを変更すると直ちに最適な状態になってしまうため，実状に則した評価が十分でない可能性がある．また，実際の Winny ネットワークでは制御対象ファイルと類似する名前を持つ制御対象外ファイルが存在する可能性があるが，本研究ではそのことを想定したシミュレーションが

できていない．そのため，より大規模で複雑な評価環境を設定し，提案手法の制御効率を再評価する必要がある．

参考文献

[1] 江崎浩，“P2P 教科書”，インプレス R&D，2008.
[2] 金子勇，“Winny の技術”，ASCII，2005.
[3] 一般社団法人コンピュータソフトウェア著作権協会:http://www2.accsjp.or.jp/（参照，調査報告書 2013.9.18）.
[4] 総務省:http://www.soumu.go.jp/ (参照，平成 24 年度 情報通信白書).
[5] 山下達也,“インターネット・トラヒック最新状況”，NTT コミュニケーションズ（2001-2010）.
[6] IIJ:http://www.iij.ad.jp/company/development/ report/iir/ (参照，Internet Infrastructure Review Vol.08).
[7] 吉田雅裕，大坐畠智，中尾彰宏，川島幸之助: “Winny ネットワークに対するインデックスポイズニングを用いたファイル流通制御方式”，情報処理学会論文誌，Vol.50，No.9，pp.2008-2022，(2009).
[8] 吉田雅裕，大坐畠智，川島幸之助: “P2P ファイル共有ネットワークにおけるファイル ID 検索に対応したポイゾニング手法の提案”，電子情報通信学会信学技法，NS2008-9 (2008-5).
[9] Liang J., Naoumov N., and Ross K.:“The Index Poisoning Attack in P2P File Sharing Systems”, Proc. IEEE Infocom 2006, pp.1-12（2006）.
[10] 吉田雅裕，大坐畠智，中尾彰宏，川島幸之助:“Share ネットワークに対するコンテンツポイズニングを用いたファイル流通制御方式”，電子情報通信学会論文誌 B，Vol.J94-B，No.5, pp.686-697,（2011）.
[11] Naoumov, N. and Ross, K.: Exploiting P2P system for DDos attacks, Proc. 1st International Conference on Scalable Information Systems，No.47，(2006).
[12] 後藤香穂，“P2P アプリケーションにおけるインデックスポイゾニングの提案”，大分工業高等専門学校卒業論文，2012.