ピアツーピアアプリケーションのセキュリティについての研究

由木 泰隆† 江崎 浩†
東京大学大学院情報理工学系研究科†

1．はじめに

現在、ピアツーピア（以後 P2P とする）と呼ばれる通信方式を利用するアプリケーションが注目を集めている。P2P とは Peer-to-Peer を意味し、各ノードが対等に接続しコミュニケーションをとる方式のことを指す。

P2P アプリケーションが注目されるきっかけとして、ファイル交換型アプリケーションの Napster[1]が挙げられる。コンテンツをサーバを介することなく、ピア同士でファイルを転送し手に入れることができる。

ファイル交換アプリケーションの普及が原因で P2P 技術は一般に知られるようになった。前述したようなネットワーク環境、ＰＣ環境の向上により、このような P2P 通信技術はファイル交換の場だけでなく様々な場面で活用されることが期待され始めており、現在でも、様々な分野に活躍の場を見出すことが出来る。

将来的には、多様なオブジェクトが P2P で結ばれたネットワーク上にあり、世界中にある PC ユーザのディスクとして利用する分散ストレージの普及が目指されており、OceanStore[6]のようなプロジェクトが存在する。分散ストレージにおけるオブジェクトの発見方法として、分散ハッシュテーブル DHT(Distributed Hash Table)を利用するものがいくつか存在する。[7][8][9][10]

本稿の構成は以下の通りである。まず、P2P ファイル交換システムにおけるリソースの発見と、DHT を利用したリソース発見について述べる。次に DHT のシステムにおいて、悪意のあるユーザが存在したときの問題点について述べ、その解決法について考察しまとめる。

2．P2P ネットワークでのリソース発見法

この章では、ファイル交換システムで現在利用されている P2P の通信を行うアプリケーションにおけるリソースの発見手法を Hybrid P2P 型と Pure P2P 型に分けて述べる。

Research on Security of Peer-to-Peer Application
† Graduate School of Information Science and Technology, The University of Tokyo.

2.1 ファイル交換

ファイル交換システムにはデータの検索をサーバに聞き、ファイル交換のみをピア同士で行う Hybrid 型や、ファイル検索もピア同士で行う Gnutella[2]に代表される Pure P2P がある。

2.1.1 Hybrid P2P

Hybrid P2P とは、望むファイルを検索するキーワードをサーバに聞きに行き、サーバから要求ファイルを所持するユーザを紹介してもらいファイルを要求するという方式であり、代表的アプリケーションとしては Napster、WinMX[5]などがある。

以下に、Hybrid P2P の代表例として、Napster における、ログイン、ファイル交換の手順を述べる。

Napster のサービスは各ユーザであるサーバント(サーバかつクライアントの意味）と Napster 社の提供するリダイレクトサーバとログインサーバという 2 種類のサーバから構成されている。リダイレクトサーバはログインサーバの負荷分散と拡張性に対する配慮のために設置されたサーバで、ログインの際にサーバントを適当なログインサーバへ振り分ける。ログインサーバはサーバントのログイン処理を行って、そのサーバントの所有している音楽ファイルのインデックスを作成するサーバである。

1.ログイン

サーバントはまず、リダイレクトサーバに対して、ログインサーバのアドレスを要求し、IP アドレスとポート番号を受け取り、指定されたログインサーバに接続し、認証を行う。

2.ファイル検索

サーバントはログインサーバへ検索要求を送信すると、ログインサーバから検索結果が返される。

3.ダウンロード

サーバントは所望ファイルを所持する相手の IP アドレス、ポート番号を受け取り、これをもとに相手サーバントとの間に接続を確立し、相手サーバントより、直接ダウンロードを開始する。

Hybrid P2P では Fig1.に示すように、各クライアントがサーバに接続し、ファイル検索を行い、サーバの返答に応じて、一般ユーザのピアへファイルのダウンロード要求を出すという形であるとまとめられる。

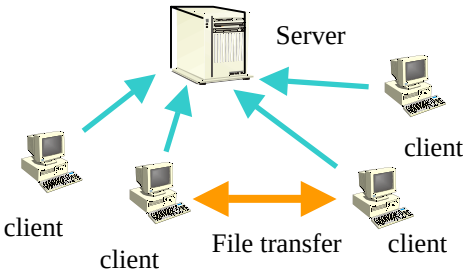

Fig.1: Hybrid P2P System

2.1.2 Pure P2P

Pure P2P とは、各ピアがお互いにメッセージを中継しあうことで、ホストの接続状況や、ファイルの検索要求などがネットワーク上で可能となるシステムである。以下に Gnutella におけるログイン手順、ファイルの検索方法を述べる。

1.ログイン

ホストキャッシュサーバを用いて知った最初の接続ノードに TCP 接続を確立し、応答のあったサーバントを GnutellaNet の入り口とする。接続が確立した後、周囲のサーバントの情報を調べるために Ping メッセージを送信する。Ping メッセージの中身は GUID(Global Unique Identifier)と TTL である。Ping メッセージは TTL の分だけ転送されて、メッセージを受け取ったサーバントは、送信されてきたときと同じ経路を用いて Pong メッセージを送り返す。以上のようにしてログインの処理は行われ、サーバントの一覧リストを得ることができる。

2.ファイル検索

検索は、直接接続しているサーバントに、Query メッセージを送信する。サーバントは Query を受け取ると、検索条件を満たすファイルがある場合は、送信されてきたときと同じ経路を用いて、QueryHit メッセージを送り返す。条件を満たすファイルがない場合は、Query/QueryHit の中継のみ行う。

3.ダウンロード

条件を満たすファイルを発見した場合、QueryHit メッセージに含まれていた情報に基づいて、相手との間に接続を確立し、ダウンロードを開始する。

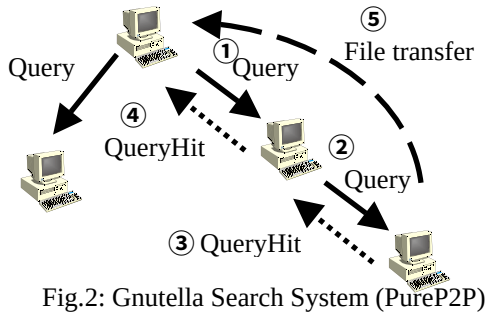

Fig.2: Gnutella Search System (PureP2P)

2.2 DHT を用いたオブジェクトの発見

多様なオブジェクトが P2P ネットワーク上にあり、世界中にあるディスク領域を仮想的に大規模のディスクとして利用してそれぞれ保存する分散ストレージという概念がある。代表的な例としてカリフォルニア大学バークレー校で行われている OceanStore[6]がある。分散ストレージにおいてはオブジェクトの発見に分散ハッシュテーブル DHT (Distributed Hash Table)を用いている。以下に OceanStore の採用している Tapestry[7]のオブジェクトを発見するメカニズムを示す。

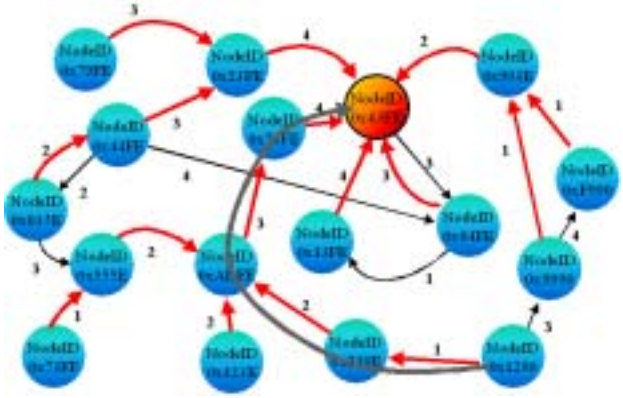
Fig.3: Tapestry Mesh

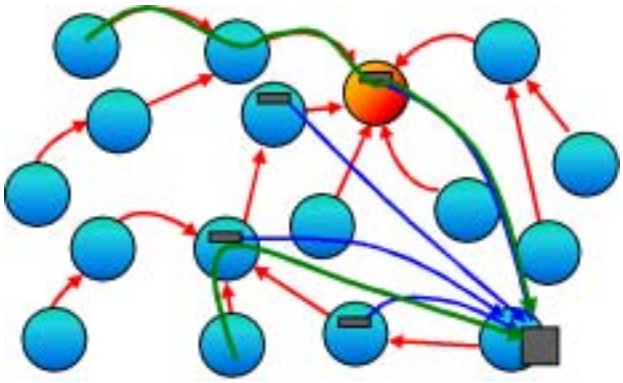
Fig.4: Object Location

Tapestry の特徴はオブジェクトのポインタを分散ストレージ上に保存する点である。

Tapestry では、Fig.3 に示すような Tapestry Mesh を作る。ノードにはそれぞれ NodeID が一意に付けられる。また、NodeID が下から何桁等しいかでルーティングテーブルを決める。各ノードは自分の NodeID の下の桁から順に i 桁一致するノードを、L_{i+1} のノードとして認識し、ルーティングテーブルを構築する。

次に、オブジェクトの登録方法について述べる。Fig.3 の右下の Node ID=1290 のノードが GUID=33FE のオブジェクトを登録することを考える。こ

のノードは[GUID=33FE,Node ID=1290]という、GUIDが33FEのオブジェクトはNode IDが1290のノードが所持しているというポインタを Node ID=33FE のノード(ルートノード)に向けて送る。このため、まず隣接した、Node ID=239E のノードに送る。このメッセージはGUIDと一致する桁が下のほうから1ずつ増えていくように、転送されていく。つまり、Fig.3なら、

1290 → 239E → ABFE → 73FE → 43FE

というノードを経由し、ポインタを配置することになる。なお、Node ID= 33FE のノードが存在しないため、GUID 33FE に一番近い Node ID=43FE のノードがこの場合のルートノードとなる。

Tapestry において、オブジェクトを発見するときは、以下の通りである。Fig.4 にあるように Node ID=1290 の持っている GUID=33FE というコンテンツへのポインタが、Node ID=239E ,ABFE , 73FE , 43FE に配置されている。例えば Fig.3,Fig.4 で左上にある Node ID=79FE のノードが GUID=33FE を探す場合、このオブジェクトのルートノード(Node ID=33FE)に向けてクエリを、まず隣接しているノード 23FE に送信する。その後、上述したルーティング方式により、

79FE → 23FE → 43FE

と転送され、ノード 43FE において、[GUID=33FE, Node ID=1290]というポインタを見つけるため、ノード 1290 に所望のオブジェクトがあることがわかる。同様に、Fig.3, Fig.4 の下方左から 2 番目にあるノード 423E が GUID=33FE のオブジェクトを探す場合も同様のルーティング形式によって、隣接したノード ABFE に[GUID=33FE,Node ID=1290]というポインタを発見できるため、やはりノード 1290 にたどり着くことができる。

3．P2P アプリケーションにおけるセキュリティ

前章で述べたように、将来的には、特定のサーバに頼らない広大な P2P ネットワークが形成されると考えられる。特に、前述した、DHT を用いるシステムは将来的に活用が期待されている。これらのシステムでは、全ての参加ノードが悪意を持たず、正常に動作することを仮定して設計されている。本章では、このシステムに悪意のあるユーザが存在したときにおこるセキュリティ的な問題を挙げる。

まず、悪意のあるユーザからの攻撃を検知する必要がある。但し、システムのプロトコルを守っていないユーザが存在したとしても、そのユーザが、本当に悪意を持っている場合と、気付かずにプロトコルを守っていない場合があるため、悪意があるユーザを確実に特定するのは難しい。

システムには、プロトコルを守っていないユーザを正しく修正する機構が求められる。

以下に具体的に想定される攻撃パターンを挙げる。

3.1 不正なクエリルーティング

悪意のあるユーザがいた場合、検索クエリを不正な（本来転送すべきでない）ノードに転送することにより、適正な検索を行うことができなくなる。これらの不正なノードは一見普通にネットワークに参加しているので、ルーティングテーブルから外れることはない。すると、検索が失敗し、クエリが悪意のあるユーザに再送されることになる。但し、このような不正なフォワーディングが露骨に繰り返された場合は、それを検出するのは容易である。不正フォワーディングが起こらないためには、クエリを投げたユーザがどのようにクエリが転送されていくかを観測できるようにすることが必要となる。

3.2 詐称問題

前章で述べたように、Tapestry 等では、ノードとオブジェクトはあるキーにマップされるが、悪意のあるユーザが、関係のないノードをあるキーにマップされているノードだと宣言すると、検索クエリを出したユーザは発行相手を間違えることになってしまう。

対応策としては次のようなものが考えられる。クエリ発行者は宛先ノードが自分宛のクエリだと認識しているか、確かめる。これにより、間に存在する悪意のノードによって詐称される問題を解決できる。

3.3 不正なルーティングテーブル問題

前章で述べたように、Tapestry 等において各ノードは周辺ノードの情報をもとに、独自のルーティングテーブルを構成する。そのため、悪意のあるユーザが存在すると、近隣のユーザに不正な情報を流すことによりルーティングテーブル構築を邪魔する可能性が生じる。これにより、一般ユーザはクエリを誤った、もしくは存在しないノードにクエリを投げることになることになってしまう。

対応策としては、不正なルーティングテーブルのアップデートを避けるために、ルーティング更新情報を検証可能なものにすることが考えられる。

3.4 ブートストラップ問題

中央にサーバの存在しない P2P システムでは最初の接続点に接続するブートストラップ機構が必要である。悪意のあるノードが複数存在し、それらが正

しいプロトコルを用いて、一見正常に見えるネットワークを構成していた場合、偶然そのネットワークに接続した善意のノードは、意図した通りの検索等を行えなくなってしまう。

この対応策としては、信頼できるソースを利用して接続することがまず挙げられる。他には、ノードが過去の接続履歴を保存し、接続履歴にある複数ノードに、ランダムなクエリを投げ、帰ってきたそれらの検索結果の一貫性を調べることにより、正しいネットワークに加入できたか調べることも、対応策として考えられる。

3.5　データ所持の詐称問題

悪意のあるノードが、一見普通にネットワークに参加している場合、この悪意あるノードは、自分の所持するオブジェクトに対する検索クエリに対して、持っていないと詐称することができてしまう可能性がある。

あるオブジェクトを単一のノードのみが持つ場合にこの問題は深刻化するため、この対応策としては、データを複製して P2P ネットワークに存在させることが挙げられる。

3.6　パケットの大量送付による DoS 攻撃

P2P ネットワークの中に存在する、ある特定のノードに対して、大量のパケットを送信し続けることにより、ターゲットのノードのサービスを停止させる DoS(Denial of Service)攻撃が行われる可能性もある。

対応策として、ひとつのノードが攻撃されても大丈夫なように、データの複製を物理的に離れた場所に配置するようにするなどが考えられる。

3.7　不必要な join と leave によるトラフィック増加

ノードが P2P ネットワークに加入(join)または脱退(leave)するときには、データの移動などをすることにより、P2P ネットワークに流れるデータ量が増大する。このことを悪意のあるユーザが利用し、不必要な join と leave を繰り返すことにより、ネットワークに無駄なデータを流すことになってしまう。

この対応策としては、悪意のあるユーザがいる場合でも、そうでない場合でも、データの移動、複製がネットワークにかける負荷を減らすシステムを設計することが考えられる。

４．終わりに

本稿では、P2P アプリケーションにおけるセキュリティ問題として、主に、分散ハッシュテーブルを用いた P2P ネットワークにおける種々のセキュリティ問題について考察した。

今後の方針としては、これら P2P アプリケーションにおけるセキュリティ強化のための手法の提案および、開発が挙げられる。

参考文献

[1] Napster　http://www.napster.com/

[2] Gnutella　http://gnutella.wego.com/

[3] Freenet
http://freenetproject.org/cgi-bin/twiki/view/JA/WebHome/

[4] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W.Hong. FreeNet: A Distributed Anonymous Information Storage and Retrieval System Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability, June 2000.

[5] WinMX　http://winmx.com/

[6] OceanStore http://oceanstore.cs.berkeley.edu/

[7] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. U. C. Berkeley Technical Report UCB//CSD-01-1141, April 2000.

[8] Antony Rowstron, Peter Druschel. Pastry:Scalable, distributed object location and routing for large-scale peer-to-peer systems Lecture Notes in Computer Science ,2001

[9] Ion Stoica and Robert Morris and David Karger and M. Frans Kaashoek and Hari Balakrishnan Chord: A scalable peer-to-peer lookup service for internet applications Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications. , 2001

[10] Sylvia Ratnasamy and Paul Francis and Mark Handley and Richard Karp and Scott Schenker　A scalable content-addressable network　Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications.

[11] Dejan S.Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, Zhichen Xu.
Peer-to-Peer Computing Technical Report
HPL-2002-57, HP Lab, 2002