2006 年度 卒業論文

Symmetric NAT に対する UDP Multi Hole Punching の技法

提出日： 2007 年 2 月 5 日

指導：後藤滋樹教授

早稲田大学　理工学部　コンピュータ・ネットワーク工学科
学籍番号： 1G03R033-9

魏　元

目次

図一覧

表一覧

第 1 章

序論

1.1 研究の背景

インターネットアクセス回線のブロードバンド化、常時接続化に伴い、インターネット通信の主流であるクライアントサーバモデルのほかに P2P モデルの通信形態が増えてきた。

P2P は違法ファイル交換問題などで悪いイメージをもたれているが、負荷分散とネットワーク全体の利用効率を向上させることができる。現在 Skype や新 napstar や Yahoo! テレビなど大企業が作る P2P ソフトも広がりつつある。

またクライアントサーバモデルのようにサーバからだけの情報発信と違い、 P2P モデルではサーバ機能を持たないマシンでも情報発信が行えるので、クライアントサーバモデルより多種多様な情報通信ができる。現にここ数年のトラフィック量は全体の 50% [1]とクライアントサーバモデルに迫りつつある。

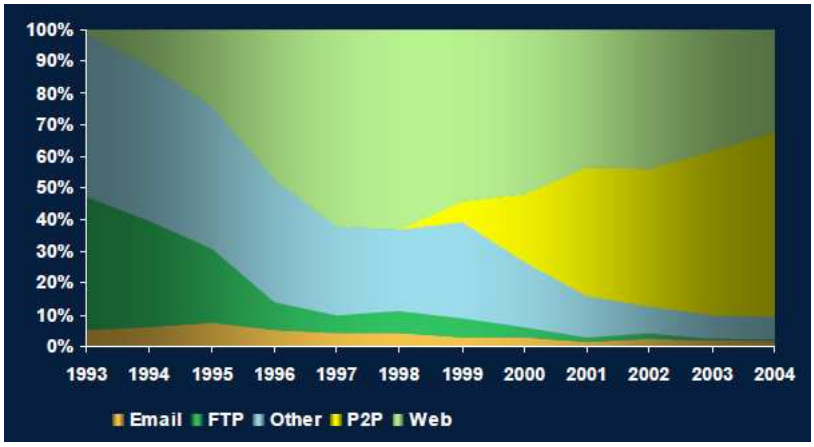


図 1.1: 最近のトラフィックの割合

[1]CacheLogic “ P2P in 2005, ” (9/05).

1.2 研究の目的

P2P モデルはネットワーク全体の利用効率が高く、現代の大容量通信に適しているが、一般的な家庭ではルータの機能（NAT）のために P2P 通信するには多くの設定が必要である。 P2P 通信は多くの場合に、ある家庭とある家庭の通信であり、一般的なユーザにとってルータの設定は難しい。そこでルータの設定なしに P2P 通信できる手法 (NAT 越えと言う) が研究されている。

現在いくつかの NAT 越えの方法が提案されているが、本論文ではその中で最も NAT 越えが難しいとされている SymmetricNAT について研究した。本論文では NAT のポートマッピングを予測し、通信するポートを多数開けることによって、両端末が NAT 下の状況で通信が成功する確率を向上させる新しい方法を提案して実装した。これを UDP Multi Hole Punching と称し、その特性を評価・考察する。

1.3 本論文の構成

本論文は以下の章により構成される。

第 1章 序論

本研究の概要について述べる。

第 2章 P2P

P2P (Peer to Peer) について解説する。

第 3章 NAT

本研究の対象であるルータの NAT 機能 (Network Address Translation) について解説する。

第 4章 従来の NAT 越えの技術

従来の NAT 越え技術について解説、その問題点について述べる。

第 5章 提案手法

本論文の提案手法である UDP Multi Hole Punching について述べる。

第 6章 実証実験

提案手法の動作検証と比較を目的とした実験と考察を行う。

第 7章 結論

本論文をまとめる。

第 2 章

P2P

2.1 P2P の特徴

P2P 通信では、従来のクライアントサーバーモデルと違い、情報資源を一括管理する固定的なサーバを設置せず、それぞれのエンドホスト（ピアと呼ばれる）が直接的な相互通信を介して自律分散的に動作する。それによって従来のサーバークライアント型の通信モデルで問題となるサーバやネットワークの負荷集中を避けることができる。

2.1.1 P2P のメリット

処理の負荷分散

クライアントサーバモデルの場合、トラフィックが増えればそれに見合う処理能力のサーバを用意しなければいけない。 P2P モデルではネットワークに参加する個々のピアが負荷を負うので、トラフィックの増加に耐えられる。

ネットワークの負荷分散

クライアントサーバモデルの場合には、ファイルのダウンロードは必ずそのファイルを持っているサーバまで問い合わせないといけない。 P2P モデルではネットワークに参加する個々のピアがファイルの断片を持っている場合があるので、ファイルの断片を持っている最寄のピアからファイルを集めることによって、ネットワークの負荷を抑えることができる。

低コスト

今まではサーバでしか情報を配信できなかったが、 P2P モデルではネットワークに参加するピアも情報発信が可能となる。参加するピア増加に伴って、トラフィックが増加するが、上記の

負荷分散によって追加設備投資をほとんどせずにネットワークを維持できる。その結果、 Napster や Skype などの小さな企業でも、大企業に匹敵するネットワークサービスを提供することができる。

2.1.2 P2P のデメリット

セキュリティ

P2P はネットワークに参加するピアも情報発信が可能なので、 Winny のウィルスである Antinny のように一度ウィルスにかかると、ウィルスのコピーや PC 内のファイルが P2P ネットワーク上に発信され取り返しのつかないことになる。

また常時ポートが開いていることや、悪意のあるスーパーノードが現れた場合には、情報が改ざんされたり、場合によっては盗聴される可能性もある。

このように P2P ソフトを使う場合には、普段のインターネット通信以上にセキュリティーに気をつける必要がある。

NAT 越えがしにくい環境

日本の場合、 ADSL や FTTH などのブロードバンドが普及し、 IP 電話とセットで契約している世帯が多い。 IP 電話とのセットで契約場合、ルータが配布され NAT 下での通信を余儀なくされる。

しかし P2P 通信ではグローバル IP アドレスでの通信が前提であり、 P2P 通信ができなかったり、本来のパフォーマンスを出せなかったりする。そのためにルータの設定を変えなくてはいけない。さらにスーパーノードを形成するピュア P2P 型の通信の場合、 NAT 下のマシンはスーパーノードになりえないので、スーパーノードとピアの割合に偏りがでてしまう。その結果スーパーノードにトラフィックと処理の負荷が集中する可能性がある。

2.2 P2P の通信形態

P2P の通信形態はいくつかあるが、ここではよく比較されるクライアントサーバモデルと二つの P2P 通信形態について解説する。新しい P2P ソフトの例もいくつか紹介する。

2.2.1 クライアントサーバモデル

クライアントサーバモデルはインターネット上の通信の基本モデルである。サーバとクライアントの 2 種類のコンピュータで通信し、サーバがポート番号で決まったサービスを提供し、クライアントが必要な時にサーバに要求を送りサービスを受け取る。現在 WEB やメールなどほとん

どのサービスでクライアントサーバモデルが使われているが、サーバの処理能力やネットワークの帯域などのボトルネックがあり、大容量な映像の配信などには向いていない。

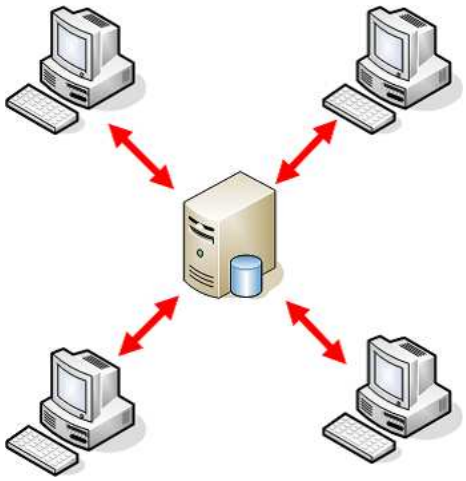


図 2.1: Client-Server

2.2.2 ハイブリッド P2P

ハイブリッド P2P 型は Napster や MSN Messenger などの主要 IM (Instant Messenger) で使われている。中心サーバにインデックス情報だけを所有し、検索と発見の仕事はサーバが担い、データのやりとりは P2P で行う方式である。

Napster では音楽ファイルを誰が保有しているかという情報をすべて中心サーバが保有しており、その情報に基づいて実際の音楽ファイルを保有者から直接入手するシステムであった。 Napster のサーバの検索をする部分が違法行為と断定され、その後登場するファイル交換ソフトは、一様に検索部分をサーバで持たないピュア P2P 型へと進化した。しかしハイブリッド P2P 型は中心サーバでファイルの情報を管理でき、負荷分散効果もあるので、ビジネスで P2P モデルを構築する場合には向いている方式である。

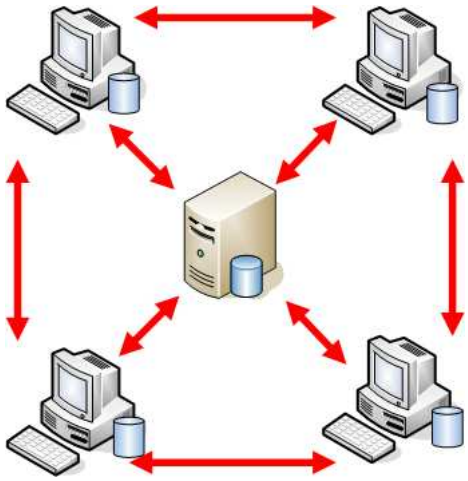


図 2.2: Hybrid P2P

2.2.3 ピュア P2P

ピュア P2P 型はサーバを持たない完全に分散している P2P 方式である。 Gnutella、 Winny、Skype などが代表的なソフトで、インデックス情報の一部を個々のピアで保持し、探したい情報をつながっているピアから順々に伝言ゲームのように聞いていき、その情報を持っていたピアは、探していたピアに通知する方式である。

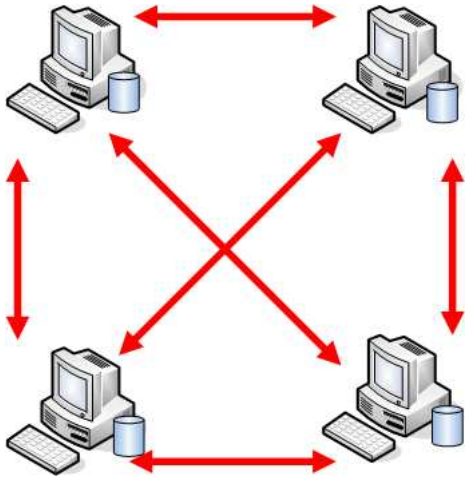


図 2.3: Pure P2P

インデックス情報を個々のピアで持っているので、ハイブリッド P2P よりも検索能力が劣る。

また Skype のように、グローバル IP アドレスを持ち、マシンスペックの高いピアをスーパーノードと定め、インデックス情報を大量に管理したり、データの中継をしたりするハイブリッド P2P 型の機能を取り入れた改良型もある。

2.2.4 P2P ソフト

BitTorrent

BitTorrent （ビットトレント）は、 Bram Cohen によって開発された、 P2P を用いたファイル転送用プロトコル及びその通信を行うソフトウェアである。「急流のように速く（ファイルを）落とせる」という意味を持つ。

数百 MB 〜 GB 単位のファイルを大勢の人に配布するために作られたシステムで、 Winny と同じく「ピアからファイルの一部を受けとるには、自分もファイルの一部を渡さなければならない」という規則に則って、アップロードする量が多ければ、ダウンロードする速度も速い仕組みになっている。 Winny が匿名性を追求したのに対して BitTorrent は匿名性は担保しない。インデックスファイルを WEB などからダウンロードし、クライアントソフトを起動させることによって通信が始まる。

現在 WEB 上にはいくつものインデックスサイトが存在し、検索がしやすいことから欧米や中国で人気を博している。さらに配信コスト低減の効果のため CentOS や LindowsOS などのリナックスディストリビューションも配布インフラとして BitTorrent を利用している。

BB ブロードキャスト

BB ブロードキャストは TV バンク社の PtoP 技術を使った動画配信システムであり、オーバーレイマルチキャスト（ OLM ）と呼ばれる P2P ベースの技術を利用する。これは専用アプリケーションをインストールした PC が、サーバから送信されたデータをほかの PC に次々に転送することで多くのユーザーにデータを配信する技術である。既存の配信技術に比べて大規模な設備投資が要らず、複数の異なる ISP を経由して配信できることから配信コストを抑えられるというメリットがある。

Skype

Skype は P2P 技術を使った IP 電話ソフトでダウンロード数 4 億 7 千万、同時オンラインユーザ数８００万人を超える、最も人気の高い IP 電話ソフトである。地域電話会社と協力することによって、有料ながら世界のほぼすべての固定電話に安く電話をかける事ができる。ピュア P2P 型を改良した方式で、グローバル IP アドレス持ち、マシンスペックの高いピアをスーパーノードとしての役割を与えることによって、インデックス情報の管理と NAT 越えできないピアへのパケットの中継をする。この方式はピュア P2P の低コスト性とハイブリッド P2P の検索効率性をあわせた方式である。

NAT 越えの方式は STUN を使用していると思われ [13]、それによって NAT 越えができない場合には、スーパーノードに UDP または TCP の音声パケットを中継 (relay) してもらう。音声

コーデックについては、 iSAC、 iLBC、非公開のコーデック、の 3 種類を自動的に選択し、音質は iSAC が最も音質が良く、 iLBC、非公開のコーデック、の順で音質が悪くなる。特に iSAC は固定電話コーデックの PMC と比べて、 PMC のデジタル化する周波数帯が 300Hz 〜 3.4kHz なのに対し、 iSAC は 50Hz 〜 8kHz の範囲をデジタル化するので、固定電話よりも臨場感があると言われる [5]。また Skype にはアナライザー機能あり、 ver2.5 の場合、 [ツール] → [設定] → [詳細] の中の [その他：通話情報を表示] にチェックをすると見ることができ、現在のコーデック、ジッター、 P2P 通信であるかあるいはパケット中継を行っているのかなどの情報がわかる。図 2.4 に Skype が通信中の様子を示す。

図 2.4: Skype 通信中のアナライザー画面

第 3 章

NAT

3.1 NAT の特徴

NAT（Network Address Translator）はネットワークアドレス変換とも呼ばれる、インターネット上のパケットに付いている IP アドレスやポート番号を別のものに変換する技術である。主にプライベート IP アドレスをインターネット上で用いるために利用される。

3.1.1 IP アドレスの枯渇

インターネットに接続されるマシンが著しく増え、さらに非効率的な IP アドレスの割り当てのために約 43 億個しかない IP アドレスは枯渇するようになった。その問題に対応（先延ばし）するために「ローカルなネットワーク内にあるホストにはプライベート IP アドレスを割り当て、インターネットに接続するときにだけグローバル IP アドレスを使うようにする」ための技術が考え出され、それがもともとの NAT である。

IP アドレス枯渇の問題は、アメリカ以外の国、例えば日本などでは割り当て IP アドレス数に比べてコンピュータの普及が著しいため、 NAT は特に重要性が高いといえる。日本向けに発売されているルータ（ADSL や FTTH などに対応したものなど）では、業務用、家庭用を問わず標準的な機能として NAT を持っていることが多い。

3.1.2 NAPT

NAPT (Network Address Port Translator) は NAT と異なり TCP/UDP のポート番号まで動的に変換されるため、一つのグローバル IP アドレスで複数のマシンから同時に接続することが可能にする技術である。 Linux に実装された IP マスカレードがポート番号まで変換することから NAPT と言われるようになった。現在、販売されているほぼすべてのルータには NAPT の機能が搭載されているので、通常使われる「NAT」と言う単語は、「NAPT」の機能も含んだものと解釈される。

3.1.3 セキュリティ

グローバル IP アドレスからプライベート IP アドレスに変換するときに、パケットフィルタリングができるため、 NAT にはセキュリティを高める効果もある。実際には、プライベート IP アドレスを割り当てられたホストには、特別な設定をしない限り外部のネットワークからは接続できないことが多い。こうした特徴から、 NAT は簡易的なファイアウォールの一種と考えることもできる。

IPv4 から IPv6 に移ると IP アドレスの枯渇問題とともに NAT の本来の意義は無くなるが、セキュリティ意識の高まりから、 NAT をファイヤーウォールを実現する安価な方法として、依然として使用されることが考えられている。

3.1.4 問題点

NAT を使ったネットワークは従来のクライアントサーバモデルでは問題なく機能する。しかし NAT 下のマシンでは P2P 通信できないという大きな問題がある。本研究では NAT 下のマシンが P2P 通信できない問題を解決するために「NAT 越え」の研究に取り組んだ。

3.2 NAT の分類

NAT は仕様や実装に関する詳細な取り決めがされていない、そのためメーカーや製品によって挙動がバラバラである． RFC3489 (STUN)[9] では、ポートのマッピングとフィルタリングの機能から、 NAT を 4 つに分類する方法が書かれている。ここではその 4 つについて説明し、さらに ”NAT Behavioral Requirements for Unicast UDP”[10] で記されたより詳細な分類を説明する。

3.2.1 Full Cone NAT

プライベートネットワーク内のホストのポートとルータのポートが 1:1 でマップされるタイプである。一度開いたポートはどこからのパケットでも受け取れることができる。

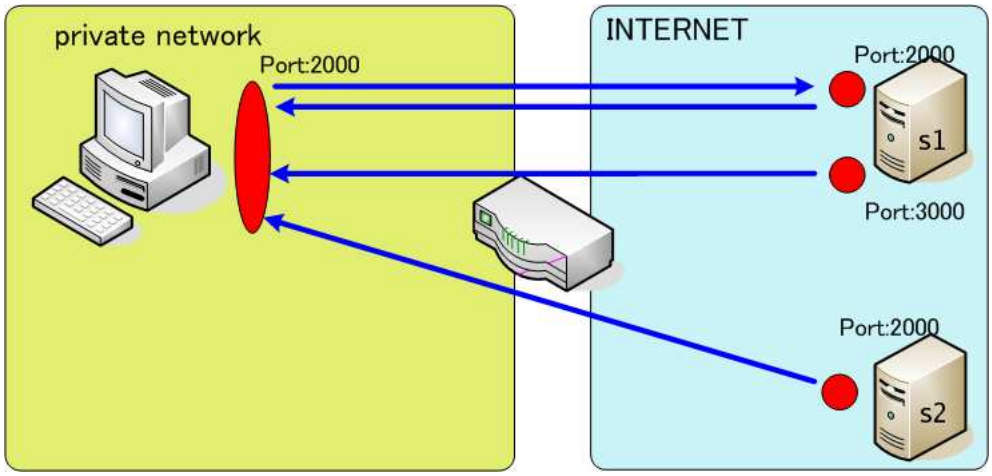


図 3.1: Full Cone NAT

3.2.2 Restricted Cone NAT

プライベートネットワーク内のホストのポートとルータのポートが 1:1 でマップされるタイプである。プライベートネットワーク内のホストから外部ネットワークに送ったことのある IP アドレスからのパケットのみを通す。図 3.2 の場合、 s2 のサーバからのパケットは通さない。

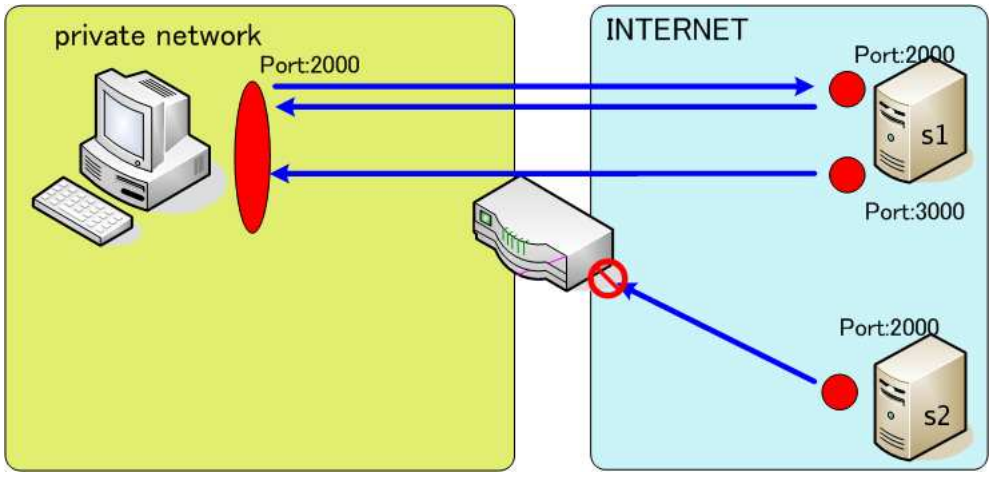


図 3.2: Restricted Cone NAT

3.2.3 Port Restricted Cone NAT

プライベートネットワーク内のホストのポートとルータのポートが 1:1 でマップされるタイプである。プライベートネットワーク内のホストから外部ネットワークに送ったことのある IP アドレスとその宛先ポートからのパケットのみを通す。

図 3.3 の場合、 s2 のサーバと s1 の 3000 番ポートからのパケットは通さない。 s1 の 3000 番ポートと通信する場合は、再びプライベートネットワーク内のホストから s1 の 3000 番ポート宛にパケットを送らないといけない。

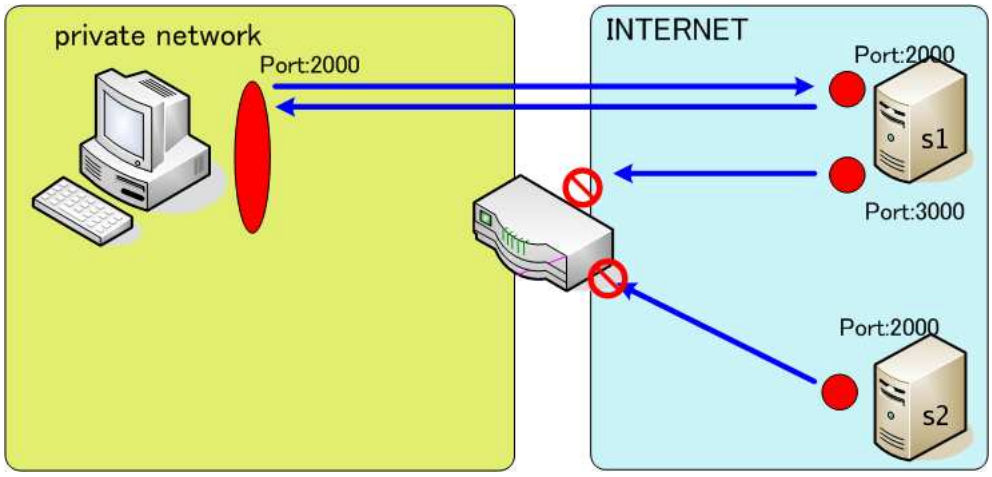


図 3.3: Port Restricted Cone NAT

3.2.4 Symmetric NAT

RFC の定義では前述の 3 つ以外の挙動をする NAT を Symmetric NAT に分類する。一般的には Poer Restricted Cone NAT と同じくプライベートネットワーク内のホストから外部ネットワークに送ったことのある IP アドレスとその宛先ポートからのパケットのみを通し、さらに同一ソケットから別の宛先に送った場合、ルータに別のポートがマップされる。これはつまり同じ宛先 IP アドレスであっても宛先ポートが違うと、別々のマッピングになることを意味する。図 3.4 の場合、プライベートネットワーク内のホストからは s1 の 2000 番と 3000 番のポートと通信しているが、 NAT は外側ネットワークにそれぞれ別ポートを割り当てる。 s1 の 3000 番ポートと通信している NAT のポートに向かって、 s2 や s1 の 2000 番ポートからパケットを送っても通らない。

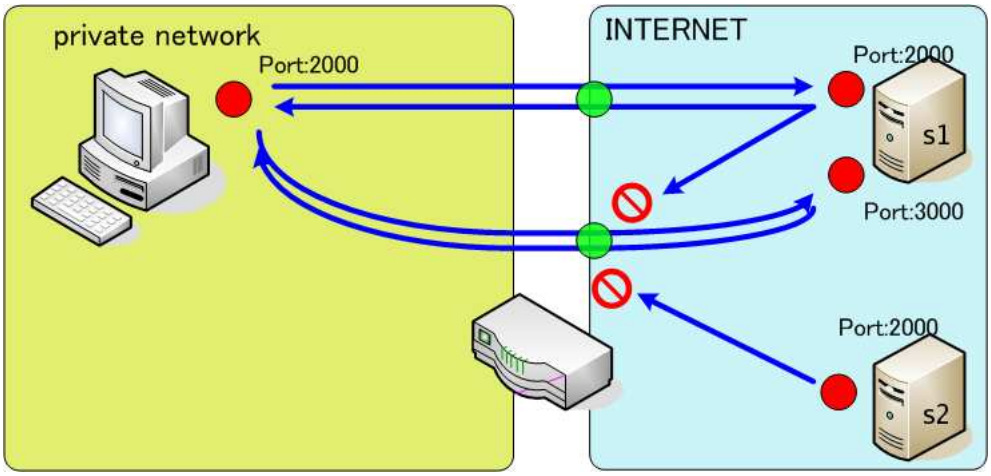


図 3.4: Symmetric NAT

3.2.5 4 種類の NAT 越えの難易度

上記から分かるように、 NAT 越えの難易度は、一度開いたポートはどこからのパケットでも受け取れることができる Full Cone NAT から、 IP アドレスを記憶する Restricted Cone NAT、IP アドレスとポートを記憶する Port Restricted Cone NAT、宛先ポート番号複数のマッピングを作る Symmetric NAT の順に難易度が上がる。

3.2.6 新しい分類法

現在 ”NAT Behavioral Requirements for Unicast UDP”[10] で記されているように、 NAT を 4 つだけの分類にするのではなく、それぞれのマッピングやフィルタリングの挙動からより細かく分類する方法が策定されている。またこの分類法は NAT 分類ソフトウェアである ”WinStun”[1] にも使われている。表 3.1 に分類項目を示す。

[1] http://sourceforge.net/projects/stun/

表 3.1: WinStun の分類表

項目	挙動結果
Nat Mapping	Independent Address Port Address and Port Connection
Endpoint Filtering	Independent Address Port Address and Port
Port number	Preserve Does not preserve
Hairpin of media	Supports Does not supports

3.3 新たなセキュリティー機能

NAT がファイヤウォールとして機能することから、メーカーではそれぞれ特徴的なセキュリティー機能を開発し、ルータ製品に実装してきた。ここでは Hole Punching を行なう時に気をつけなければならない仕様を取り上げる。

3.3.1 SPI

SPI (Stateful Packet Inspection) は現在、市販のルータにほぼすべて搭載されているセキュリティー機能である。プライベートネットワーク内から送信したデータをセッションログとして保管しておき、グローバルネットワークから到着したパケットがセッションログと矛盾しないか確認する。もし、グローバルネットワークから送信されてきたパケットとセッションログが矛盾する場合にはパケットを遮断し、プライベートネットワーク内のホストを不正アクセスから保護する機能を有する。このため一度開けたポートに別のマシンからのパケットを送って通信を成り立たせる Hole Punching 方法では、成功する確率が低い。

3.3.2 マッピングを書き換える NAT

この現象は iptables を NAT をして使用したときに発見したものである。 NAT 下のあるホストとグローバルネットワーク上のあるサーバ通信している状態で、その NAT の (NAT とサーバの通信している) ポートに向かって別の IP アドレスからパケットを送ると、 NAT がマッピングを書き換えてしまう。したがって次回グローバルネットワーク上のあるサーバにパケットを送る場合は別のポートから送り出されることになる。

このように各社によって NAT の仕様がバラバラで、さらに疑わしい方法を遮断セキュリティー機能が存在する。本論文では各社の仕様に対応すべく、より自然な通信を提案する。

第 4 章

従来の NAT 越えの技術

1.2 にも書いたように、 NAT によって P2P 通信は成り立たなくなってしまう。そこで、近年 NAT 下のホストからでも P2P 通信を可能にするために、「NAT 越え」の研究が TCP、 UDP の両方で研究されている。

TCP Hole Punching については、「NAT Blaster」、「P2PNAT」、「STUNT」などの方式が提案されている。本章では UDP Hole Punching についての代表的な手法をいくつか取り上げる。

4.1 UPnP

UPnP (Universal Plug and Play) は、 UPnP Forum で規定された規格で、 PC や AV 機器など家庭内のいろいろな機器をネットワークを通じて接続し、相互に機能を提供しあう技術である。 UPnP を利用すると、プライベートネットワーク内のホストから UPnP 対応のルータに対して、ルータのもつグローバル IP アドレスの割り当てや、ポート番号のマッピングを要求することができる。ルータに加えてホストでも UPnP に対応すれば、プライベートネットワークとグローバルネットワークとの間で、 P2P 通信することができるようになる。ただし市販のすべてのルータが UPnP に対応していないのに加えて、 Windows Messenger 以外の UPnP を使ったソフトウェアを保障していないメーカーもあり、万全とはいえない。

4.2 Teredo

Teredo （RFC 4380） はマイクロソフトが提案した、 IPv4 の UDP トンネリング技術を使い IPv6 を用いた P2P 通信を実現する方法である。 Teredo Server と呼ばれるインターネット上のサーバを仲介者とすることで、 NAT のマッピングテーブルに「穴」を開け、それを利用して UDP/IPv4 通信を実現し、その上に IPv6 トンネルを構築する方式である。しかし Symmetric

NAT には対応していない。

4.3 STUN

STUN (Simple Traversal of UDP through NATs) [9] は、アプリケーションがグローバルネットワークとの間の NAT、およびファイアウォールの存在とタイプを発見することを可能にする軽量プロトコルである (STUN 自体には P2P 通信をする機能は備わっていない)。 STUN は、NAT がプライベートネットワーク内のホストに割り当てた IP アドレスやポート番号を、アプリケーションに通知することができる。ただしこの方法を利用するためには、グローバルネットワークに STUN サーバが必要となり、プライベートネットワーク内のホストでも STUN に対応する必要がある。これに加えて、途中に介在する NAT の動作が Cone 型でなければならない。つまり、 NAT の内側の同じ IP アドレス、ポート番号から外側にパケットを送信する際に、NAT の外側に割り当てられる IP アドレス、ポート番号が常に同じである必要がある。

以下の図 4.1 に STUN を用いた通信例を示す。

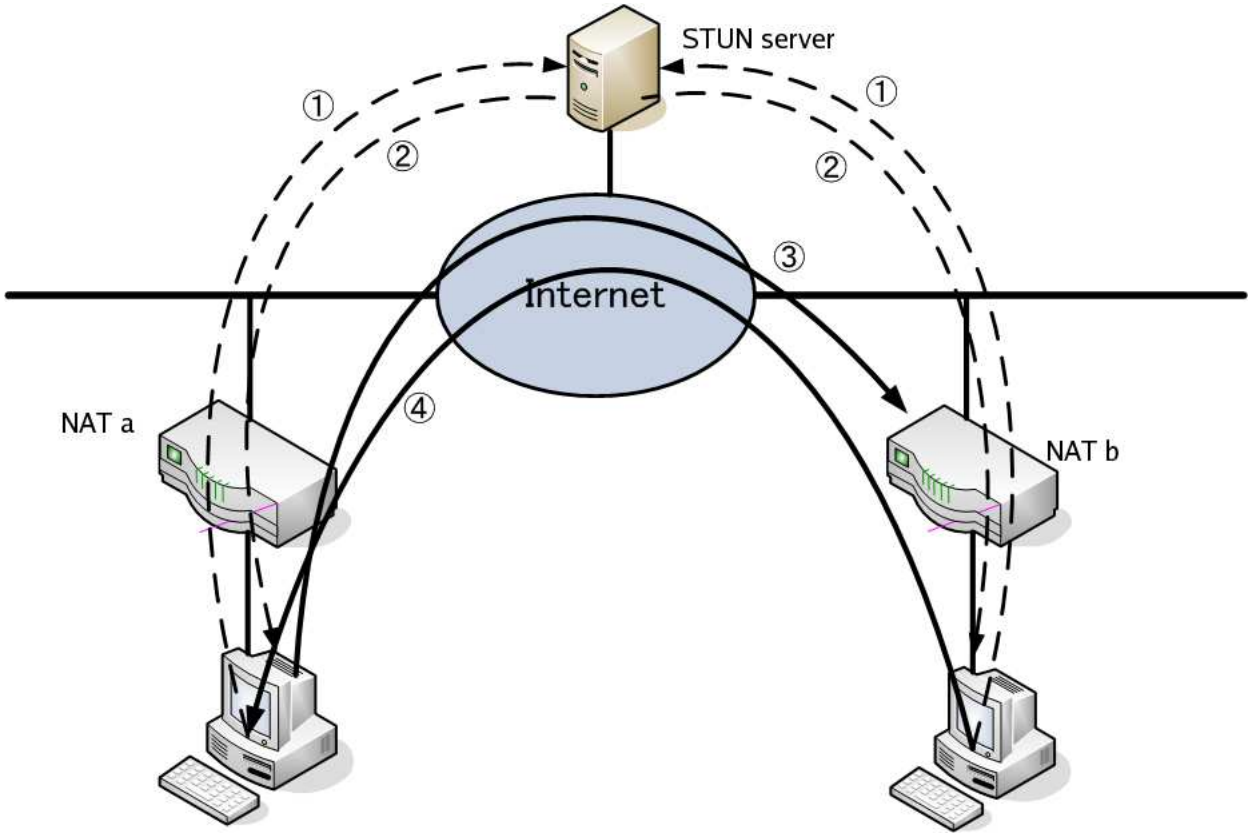


図 4.1: STUN を用いた UDP Hole Punching の例

1. 双方のホストから STUN server へパケットを送る (図 4.1 中の 1)

2. STUN server から、通信相手の IP とポート番号を教えてもらう (図 4.1 中の 2)
3. 一方ホストからもう一方のホスト (NAT b) の IP アドレスとポート番号宛にパケットを送る (図 4.1 中の 3)
4. もう一方のホストから NAT a の IP とポート番号宛にパケットを送る (図 4.1 中の 4)

図 4.1 中の 3 のパケットにより、 4 のパケットが NAT a の内側にあるホストへと届くマッピングテーブルが作られる。この方法が成功すると、 4 のパケット以降に双方の NAT のマッピングに矛盾がないので P2P 通信が成立する。

現在 UDP による「NAT 越え」は Symmetric NAT の問題や 3.3 の新たなセキュリティー機能の問題がまだ残っている。そこで第 5 章では、 NAT のセキュリティーに引っかからずに Symmetric NAT をも越えることができる手法を提案する。

第 5 章

提案手法: UDP Multi Hole Punching

5.1 概要

本論文では、2 台のサーバとの通信による port prediction と、低い TTL に設定した UDP パケットを大量に送ることによる Multi Hole Punching の方法を提案する。システムの概要を図 5.1 に示す。また、本論文の手案手法の通信シーケンスを 3 つのフェーズに分けて以下に示す。

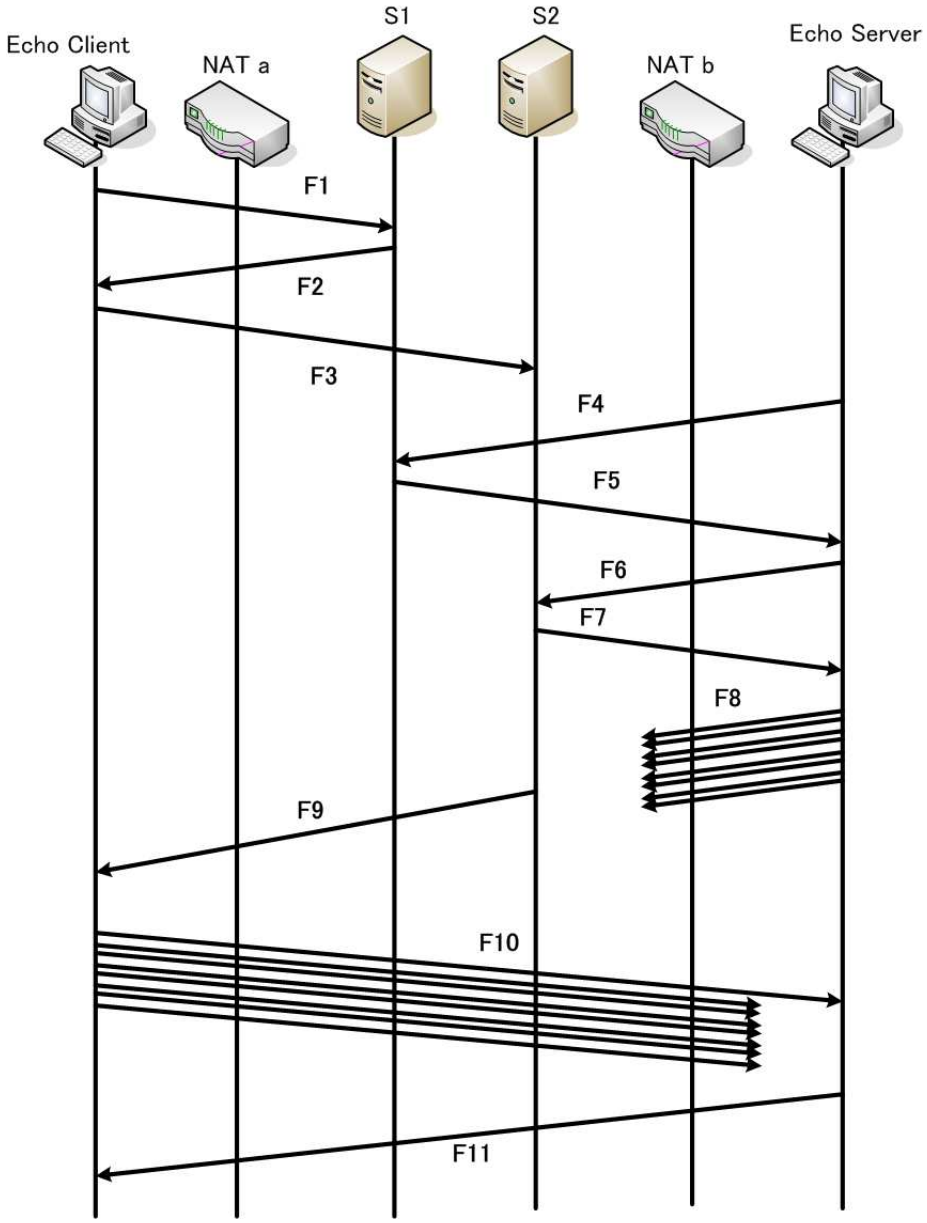


図 5.1: UDP Multi Hole Punching の概要

5.2 フェーズ I

フェーズ I では、 Echo Client がサーバと通信し、 Echo Client のグローバル IP アドレスとポートをサーバ S1、 S2 に伝える。通信の過程を図 5.2 に示す。

1. F1 では Echo Client からサーバの S1 にパケットを送る。 S1 では送られたパケットの NAT a のポートを解析する。
2. F2 では解析したポート番号を Echo Client に伝える。
3. F3 では Echo Client からサーバの S2 に、 **S1 との通信で使ったポート番号**の情報が入ったパケットを送る。 S2 では送られたパケットの NAT a のポートを解析し、そのポート番号と S1 との通信に使ったポート番号、 Echo Client のグローバル IP アドレスを記憶する。 Echo Client は待機モードに入る。

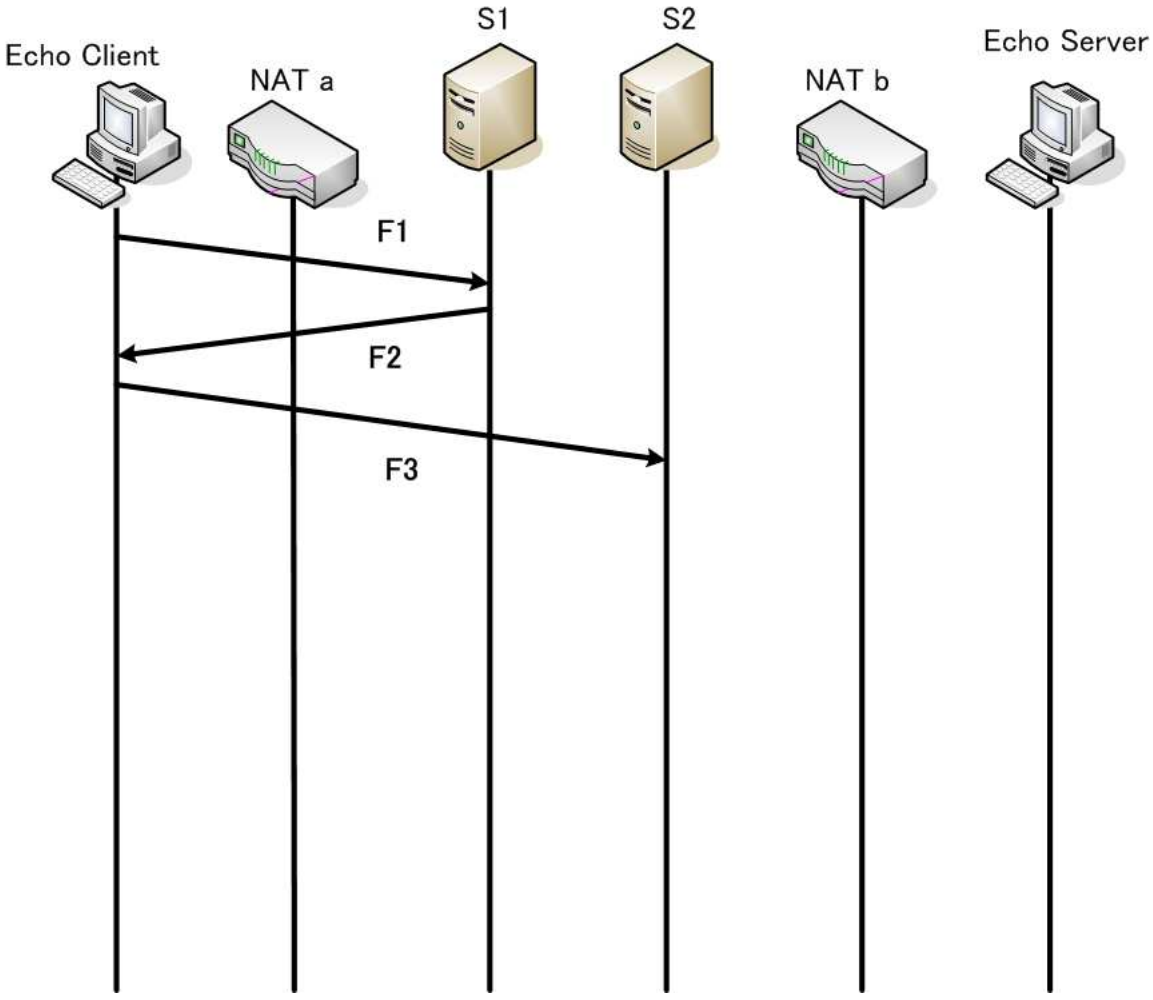


図 5.2: フェーズ I

5.3 フェーズ II

フェーズ II ではフェーズ I と同じように、 Echo Server がサーバと通信し、 Echo Server のグローバル IP アドレスとポートをサーバに伝える。通信の過程を図 5.3 に示す。

1. F4 では Echo Server からサーバの S1 にパケットを送る。 S1 では送られたパケットの NAT b のポートを解析する。
2. F5 では解析したポート番号を Echo Server に伝える。
3. F6 では Echo Server からサーバの S2 に、 **S1 との通信で使ったポート番号と通信したい IP アドレス**の情報が入ったパケットを送る。 S2 では送られたパケットの NAT b のポートを解析し、そのポート番号と S1 との通信に使ったポート番号、 Echo Server のグローバル IP アドレスを記憶する。
4. Echo Server から送られた**通信したい IP アドレス** が S2 に記憶されていた場合、フェーズ III へ進む。

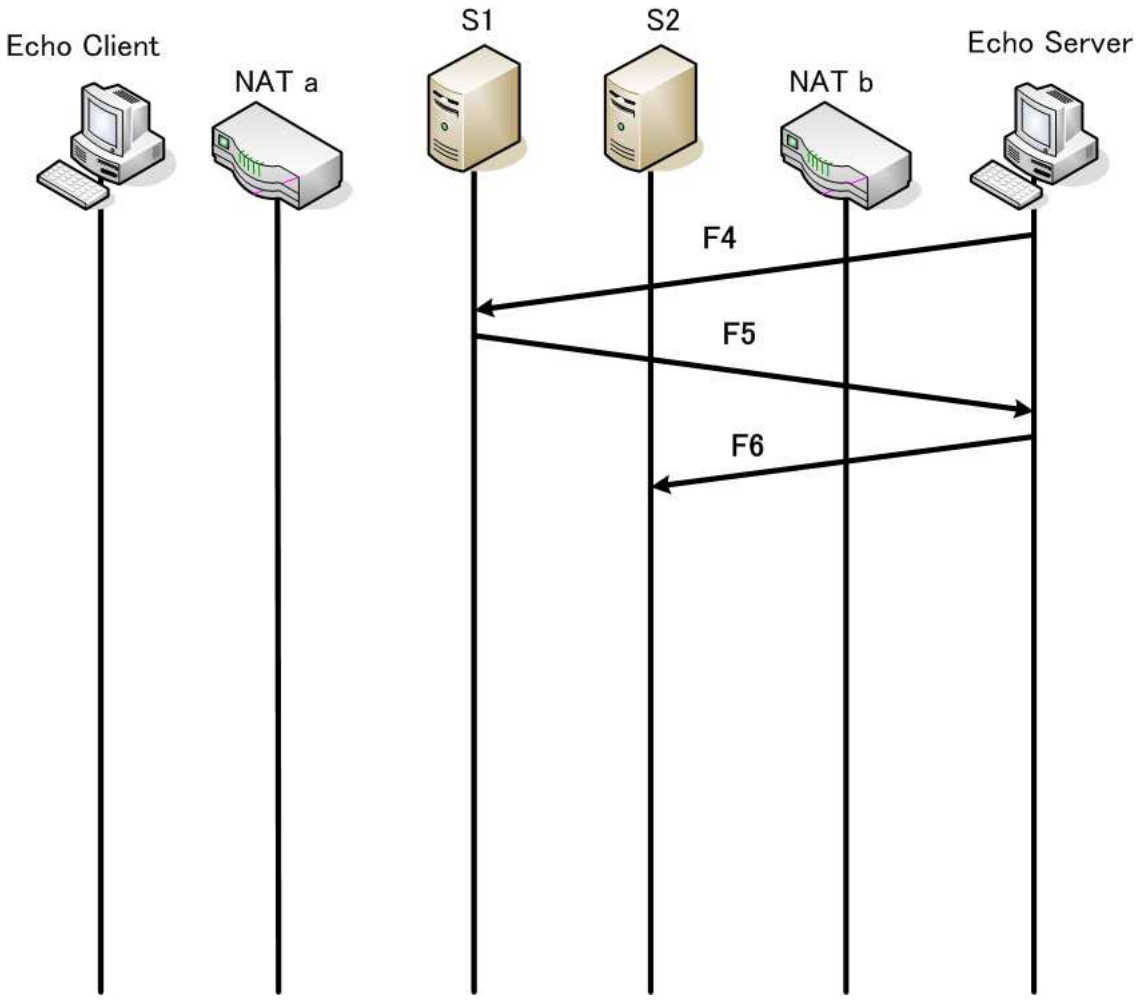


図 5.3: フェーズ II

5.4 フェーズ III

フェーズ III では送られた情報からポートの予測をし、宛先の開始ポートとポートの間隔（punching mode: ポートをインクリメントしていくのか、デクリメントしていくのか、一つ飛びにするのか）を Echo Client と Echo Server に伝える。その後 Echo Client と Echo Server が Multi Hole Punching を行って P2P 通信を成立させる。通信の過程を図 5.4 に示す。

1. F7 では **NAT a が S1 と S2 との通信に使った**ポート番号を照らし合わせる。予測された宛先の開始ポート番号とポートの間隔（punching mode）を、 S2 から Echo Server に伝える。

2. F8 では Echo Server が受けとった情報どおりに、宛先 IP アドレスと宛先の開始ポート番号とパケットの TTL を 2 〜 3 に設定する。 for 文で、宛先ポート番号を punching mode に沿って指定し、自分のソースポートをインクリメントしながら bind する。その後セットさ

れたパケットを送り出す。この操作 (Hole Punching) を 1000 回繰り返し（つまり 1000 コのポートを開ける）、それぞれが受信モードに入る。

3. F9 では F7 と同じように **NAT b が S1 と S2 との通信に使った**ポート番号を照らし合わせる。予測された宛先の開始ポート番号とポートの間隔（punching mode) を S2 から Echo Client に伝える。

4. F10 では Echo Client が受けとった情報どおりに、宛先 IP アドレス (NAT b) と宛先の開始ポート番号を設定する。 for 文で、宛先ポート番号を punching mode に沿って指定し、自分のソースポートをインクリメントしながら bind する。その後セットされたパケットを送り出す。この操作を 1000 回繰り返し（NAT b 宛てに普通の UDP パケットを送る）、それぞれが受信モードに入る。

5. その後送られた UDP パケットが NAT b に届き、 Echo Server に転送されたら、そのパケットを解析して NAT a のポート番号と、自分に届いたポート番号を記憶する。自分に届いたポート番号以外の bind したポートを close する。

6. F11 では Echo Server が記録した **NAT a のポート番号と、自分に届いたポート番号**を基に TTL を UDP パケットを作り送り返す。このパケットは NAT a が変換して送り出した、パケットと完全に整合するので、問題なく Echo Client に転送される。

7. Echo Client に届いたパケット解析して NAT a のポート番号と、自分に届いたポート番号を記憶する。自分に届いたポート番号以外の bind したポートを close する。これで Echo Client と Echo Server の間には整合性のあるルートが生まれて、 P2P 通信が成立する。

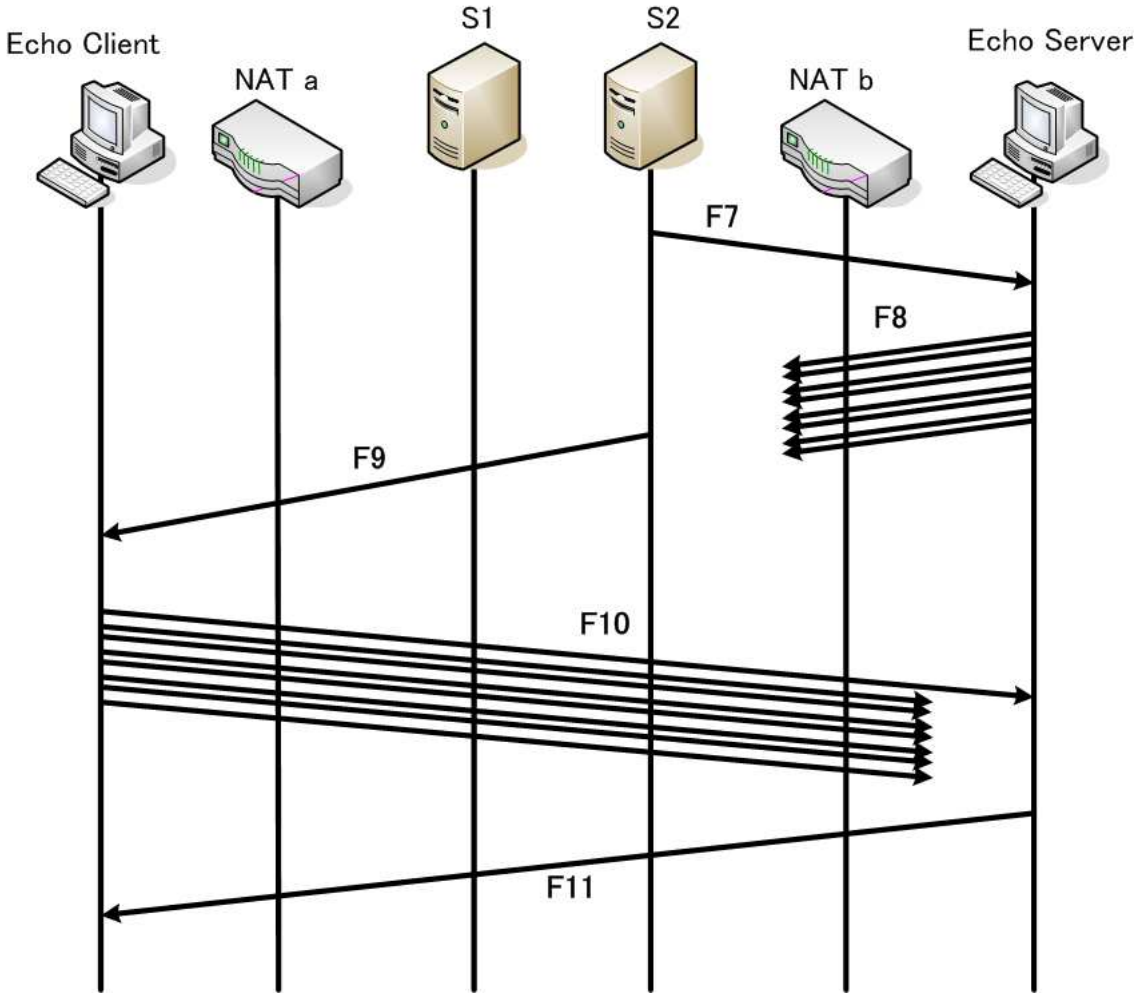


図 5.4: フェーズ III

5.5 提案手法の特徴

自然な UDP 通信

3.3 で取り上げたように、「通信中のポートに別の IP アドレスからパケットを送ると、 NAT がマッピングを書き換えてしまう」現象があった。この挙動に対しては、従来の方法であるサーバとの通信中のポートに別の IP アドレスのピアからパケットを送って NAT 越えをする方法は通用しない。

そこで提案手法では、 Hole Punching をする Echo Server 側では、 2 〜 3 の低い TTL を設定し NAT b を通り抜け、かつ NAT a に到達しないようにする。これによって、図 5.4 の NAT a および NAT b に注目すると、 NAT b については、 F8 で NAT a の IP アドレスに送り、 F10 で NAT a からパケットが返ってきた通常の UDP 通信であり、 NAT a についても F10 で NAT b

の IP アドレスに送り、それが F11 で帰ってきた通常の UDP 通信に見える。双方の NAT にとって自然な UDP 通信を装うことで、検知されにくくなる。

正確なポート予測

既存の UDP Hole Punching の手法ではマッピングを調べるサーバは一台であった。提案手法では 2 台のサーバを使う。多くの NAT のポート変換アルゴリズムはインクリメント型、デクリメント型、「2 , 4 , 6 , 8 . . . 」のような一個飛び型、その他（ランダム型）であり、 2 台のサーバを使うことによって、インクリメント型、デクリメント型、一個飛び型に対応できる。

発信ポートをコントロール

提案手法では通信に使うポートを bind してから使う。このようにすることで、 NAT のポート変換アルゴリズムがランダム型をコントロールできる場合があり、ソースポートが「x , x+1, x+2...」のように送った場合、「n , n + 1, n + 2...」のように規則正しくポート変換される。

1000 個の異なるポートから送る

1000 個の異なるポートから送りあうことで、 NAT a が予想できないポート変換をするときでも、 Echo Server 側の NAT b のマッピングと一致していれば P2P 通信が成り立つから、 NAT 越えの確率を高めることができる。

第 6 章

実証実験

6.1 実験の目的

本論文では提案手法である UDP Multi Hole Punching による Symmetric NAT 越えを目指す。まず、実験 1 (WinStun を使った NAT の分類) および実験 2 (パケットキャプチャー) で Symmetric NAT ルータを識別する。実験 3 では、提案手法との比較のために Skype を使って NAT 越えの実験をする。実験 4 では、提案手法による NAT 越えの実験をし、 Skype の結果と比較する。

なお、実験に使用したルータを表 6.1 に示す。

表 6.1: 使用したルータ

メーカー	型番
iptables	iptables
I-O DATA	NP-BBRL
I-O DATA	WN-WAPG/R
BAFFALO	BBR-4HG
LINKSYS	BEFSR41C-JP3
PRANEX	BRL-04CW
corega	CG-BARMX2
NEC	AtermBR1500H
Cisco Systems	Cisco2621

6.2 実験 1 : WinStun を使った NAT の分類

実験 1 では ”WinStun”[1]を使用して ”NAT Behavioral Requirements for Unicast UDP” [10] に基づいて NAT の特性を調査する。

本実験は図 6.1 のようなネットワーク構成とし、 PC1 にはローカル IP アドレスを割り当てた。

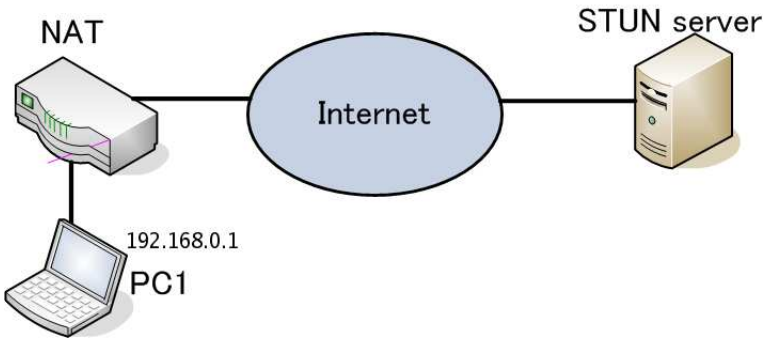


図 6.1: 実験 1 の構成

[1]http://sourceforge.net/projects/stun/

”WinStun”の使い方

”WinStun”にはサーバプログラムとクライアントプログラムが存在する。 NIC が 2 つあるサーバを STUN Server としてサーバプログラムを起動する、 WindowsOS の入っている PC1 から STUN Client のプログラムを起動すると、 NAT の挙動を検査した結果が PC1 に返ってくる。

6.3 実験 2 : パケットキャプチャー

実験 2 では、 NAT のポート変換アルゴリズムを調べるために、ソースポートを 5323 番に固定し、 bind した。その後宛先のポートを次々に変えたパケットを送り出す。そのパケットをパケットキャプチャソフト[2]を利用してキャプチャーし調査を行った。

本実験は図 6.2 のようなネットワーク構成とし、 PC1 にはローカル IP アドレスを割り当てた。また、 Switch をミラーポートモードにして、 PC2 で複製されたパケットをキャプチャーした。

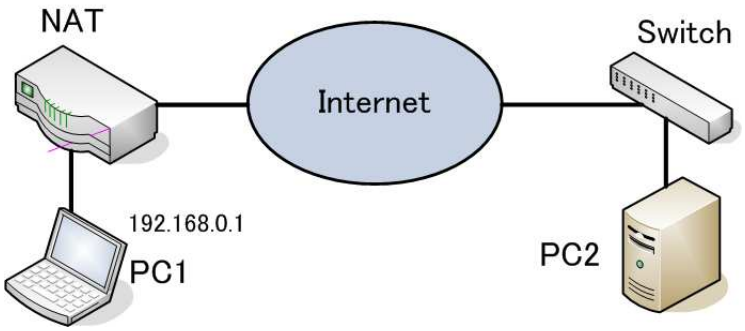


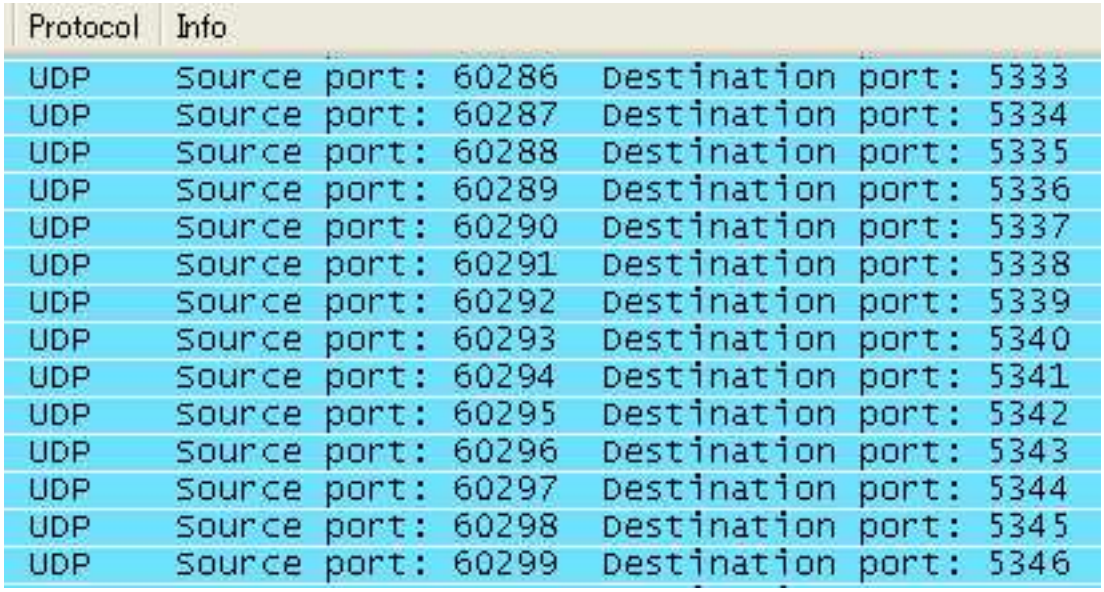
図 6.2: 実験 2 の構成

[2]Wireshark (http://www.wireshark.org/)

6.4 実験 1 および実験 2 の結果と考察

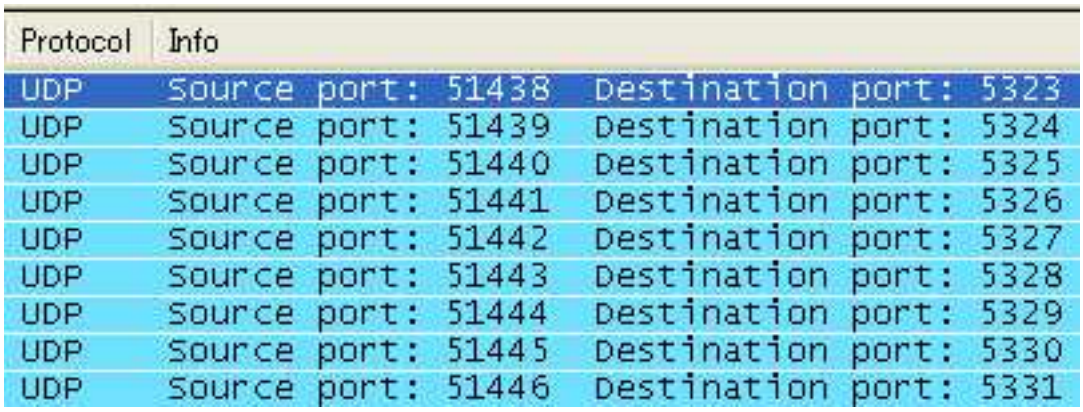
実験 1 の結果を付録 A にまとめる。

実験 2 の結果を図 6.3 、図 6.5 、および図 6.6 に示す。



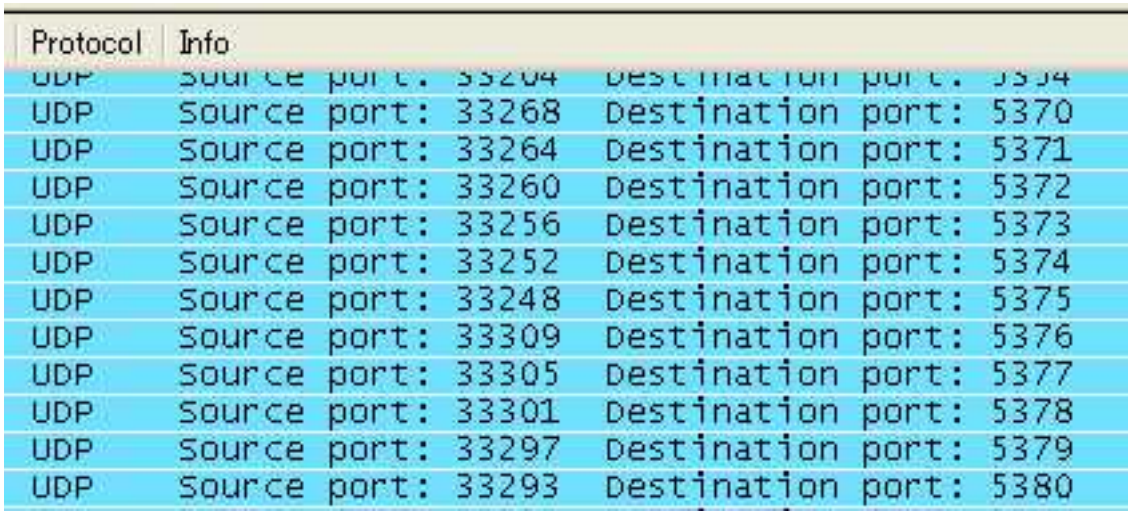
Protocol	Info
UDP	Source port: 60286 Destination port: 5333
UDP	Source port: 60287 Destination port: 5334
UDP	Source port: 60288 Destination port: 5335
UDP	Source port: 60289 Destination port: 5336
UDP	Source port: 60290 Destination port: 5337
UDP	Source port: 60291 Destination port: 5338
UDP	Source port: 60292 Destination port: 5339
UDP	Source port: 60293 Destination port: 5340
UDP	Source port: 60294 Destination port: 5341
UDP	Source port: 60295 Destination port: 5342
UDP	Source port: 60296 Destination port: 5343
UDP	Source port: 60297 Destination port: 5344
UDP	Source port: 60298 Destination port: 5345
UDP	Source port: 60299 Destination port: 5346

図 6.3: NEC ルータのキャプチャーデータ



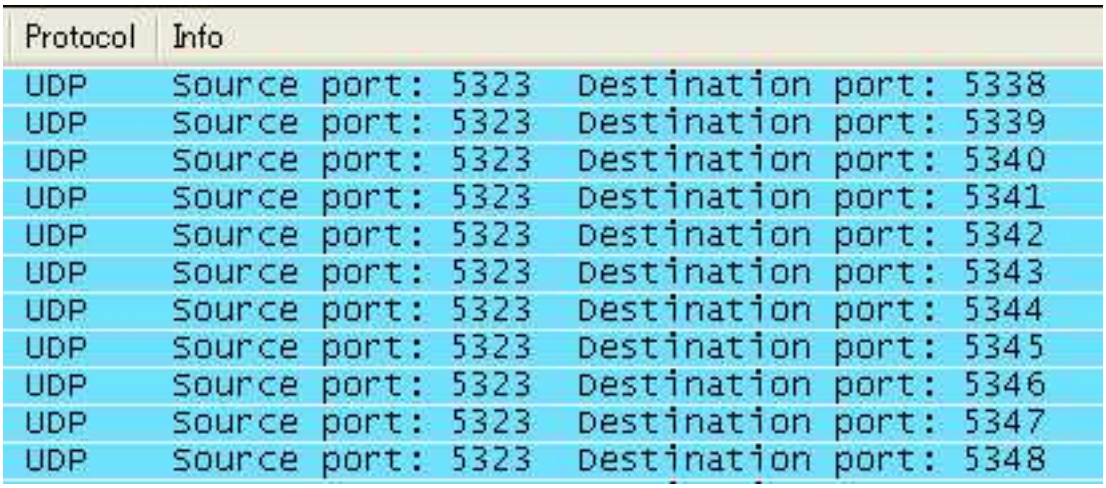
Protocol	Info
UDP	Source port: 51438 Destination port: 5323
UDP	Source port: 51439 Destination port: 5324
UDP	Source port: 51440 Destination port: 5325
UDP	Source port: 51441 Destination port: 5326
UDP	Source port: 51442 Destination port: 5327
UDP	Source port: 51443 Destination port: 5328
UDP	Source port: 51444 Destination port: 5329
UDP	Source port: 51445 Destination port: 5330
UDP	Source port: 51446 Destination port: 5331

図 6.4: I-O DATA (WN-WAPG/R) ルータのキャプチャーデータ



Protocol	Info
UDP	Source port: 33268 Destination port: 5370
UDP	Source port: 33264 Destination port: 5371
UDP	Source port: 33260 Destination port: 5372
UDP	Source port: 33256 Destination port: 5373
UDP	Source port: 33252 Destination port: 5374
UDP	Source port: 33248 Destination port: 5375
UDP	Source port: 33309 Destination port: 5376
UDP	Source port: 33305 Destination port: 5377
UDP	Source port: 33301 Destination port: 5378
UDP	Source port: 33297 Destination port: 5379
UDP	Source port: 33293 Destination port: 5380

図 6.5: PLANEX ルータのキャプチャーデータ



Protocol	Info
UDP	Source port: 5323 Destination port: 5338
UDP	Source port: 5323 Destination port: 5339
UDP	Source port: 5323 Destination port: 5340
UDP	Source port: 5323 Destination port: 5341
UDP	Source port: 5323 Destination port: 5342
UDP	Source port: 5323 Destination port: 5343
UDP	Source port: 5323 Destination port: 5344
UDP	Source port: 5323 Destination port: 5345
UDP	Source port: 5323 Destination port: 5346
UDP	Source port: 5323 Destination port: 5347
UDP	Source port: 5323 Destination port: 5348

図 6.6: それ以外のルータのキャプチャーデータ例

実験 1 では、 Cisco2621 と IO-DATA (WN-WAPG/R) については計測不能で、 NEC のルータについては ”VOIP will NOT work” というメッセージから Symmetric NAT の可能性が高い。そのほかはすべて STUN による NAT 越えができる判定結果となった。

実験 2 では、 NEC と IO-DATA (WN-WAPG/R) と PLANEX 以外のルータはすべて、図 6.6 のように宛先ポートが違うにも関わらず、ポート変換した後もソースポートがずっと同じ番号であった。このような挙動をする NAT は Symmetric NAT ではない。実験 1 では Cisco Systems のルータが測定不能であったが、実験 2 によって Cisco Systems のルータが Symmetric NAT ではないと断定できる。

NEC のルータについては (図 6.3) 、ソースポートを 5323 番に固定して送り出しているにも

関わらず、 NAT によるポート変換後のソースポートがそれぞれ違っていた。これは Symmetric NAT の挙動である。またソースポートが一番ずつ増えているので、インクリメント型のポート変換アルゴリズムを使用していることが分かる。

IO-DATA (WN-WAPG/R) については同じく (図 6.4) 、ソースポートを 5323 番に固定して送り出しているにも関わらず、 NAT によるポート変換後のソースポートがそれぞれ違っていた。これは Symmetric NAT の挙動である。またソースポートが一番ずつ増えているので、インクリメント型のポート変換アルゴリズムを使用していることが分かる。

PLANEX のルータについても同じく (図 6.5) 、ソースポートを 5323 番に固定して送り出しているにも関わらず、 NAT によるポート変換後のソースポートがそれぞれ違っていたので Symmetric NAT の挙動である。またソースポートが 4 ずつ減っているが途中で 33248 番から 33309 番に飛んでいるので、ランダム型のポート変換アルゴリズムを使用していると断定する。

6.5 実験 3: Skype による NAT 越えの実験

実験 3 では、提案手法との比較のために NAT 越えに STUN を応用していると思われる Skype [13] を使って NAT 越えの実験をする。

図 6.2 のようなネットワーク構成とし、 PC1 および PC2 にはローカル IP アドレスを割り当てた。また、 NAT a と NAT b のグローバル IP アドレスは実際の「一般家庭同士の通信」を再現するために別々のセグメントの IP アドレスを割り当てた。

さらに PC2 で Skype のアナライザーとキャプチャーソフトの Wireshark を起動し、通信を調べた。

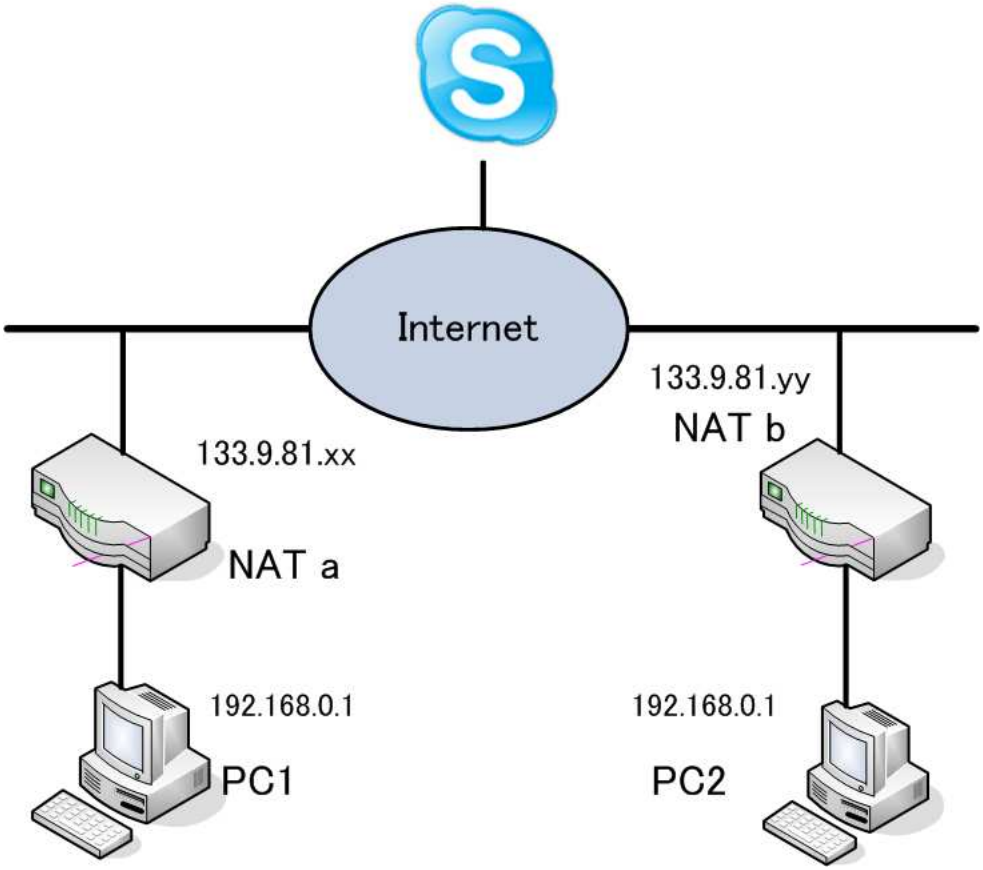

Internet
133.9.81.xx
NAT a
133.9.81.yy
NAT b
192.168.0.1
PC1
192.168.0.1
PC2

図 6.7: 実験 3 の構成

6.6 実験 3 の結果と考察

6.6.1 結果

Skype の通信には 3 種類の通信方法が存在した。その 3 種類は UDP による P2P 通信と UDP の中継通信（ UDP-RELAY ）と TCP の中継通信（ TCP-RELAY ）である。

UDP による P2P 通信をの画面を図 6.8 に示し、その通信時のキャプチャー画面を図 6.9 に示す。 P2P 通信と UDP の中継通信（ UDP-RELAY ）をの画面を図 6.10 に示し、その通信時のキャプチャー画面を図 6.11 に示す。 TCP の中継通信（ TCP-RELAY ）をの画面を図 6.12 に示し、その通信時のキャプチャー画面を図 6.13 に示す。

また各種 NAT との通信の結果を表 6.2 に示す。

図 6.8: I-O DATA と corega の組み合わせで P2P 通信中の画面

Source	Destination	Protocol
133.9.81.185	192.168.0.1	UDP
192.168.0.1	133.9.81.185	UDP
133.9.81.185	192.168.0.1	UDP
192.168.0.1	133.9.81.185	UDP
133.9.81.185	192.168.0.1	UDP
192.168.0.1	133.9.81.185	UDP
133.9.81.185	192.168.0.1	UDP
192.168.0.1	133.9.81.185	UDP

図 6.9: I-O DATA と corega の組み合わせ通信をキャプチャーした図

図 6.10: iptables と NEC の組み合わせで UDP-RELAY 通信中の画面

Source	Destination	Protocol
81.187.205.196	192.168.0.1	UDP
192.168.0.1	81.187.205.196	UDP
81.187.205.196	192.168.0.1	UDP
192.168.0.1	81.187.205.196	UDP
81.187.205.196	192.168.0.1	UDP
192.168.0.1	81.187.205.196	UDP
81.187.205.196	192.168.0.1	UDP

図 6.11: iptables と NEC の組み合わせ通信をキャプチャーした図

図 6.12: iptables と NEC の組み合わせで TCP-RELAY 通信中の画面

Source	Destination	Protocol
140.113.69.58	192.168.0.1	TCP
192.168.0.1	140.113.69.58	TCP
192.168.0.1	140.113.69.58	TCP
140.113.69.58	192.168.0.1	TCP
140.113.69.58	192.168.0.1	TCP
192.168.0.1	140.113.69.58	TCP
192.168.0.1	140.113.69.58	TCP
192.168.0.1	140.113.69.58	TCP
140.113.69.58	192.168.0.1	TCP

図 6.13: PLANEX と NEC の組み合わせ通信をキャプチャーした図

表 6.2: 実験 3 の結果 RU は Relay UDP、 RT は Relay TCP、 o は P2P 通信を表す

	iptables	I-O	I-O2	BAFFALO	LINKSYS	PRANEX	corega	NEC	Cisco
iptables	RT	RT	RU	o	o	RT	RT	RU	RT
I-O DATA (NP-BBRL)	/	/	o	o	o	RU	o	RU	o
I-O DATA (WN-WAPG/R)	/	/	/	o	o	RU	RT	RU	RT
BAFFALO	/	/	/	/	RU	RT	o	o	o
LINKSYS	/	/	/	/	/	o	o	o	o
PRANEX	/	/	/	/	/	/	RT	RT	RT
corega	/	/	/	/	/	/	/	RU	o
NEC	/	/	/	/	/	/	/	/	RT
cisco	/	/	/	/	/	/	/	/	/

表 6.2 の結果から、 Skype が積極的に P2P 通信していないことがわかった。 Skype のアナライザーの結果を見ると、すべての場合で音声品質が良かった (Q=QUALITY OK) のでパケットを中継しても、品質が下がらない場合は、 UDP Hole Punching を積極的にしないことが推測される。しかし、実験 1 、 2 で判定した NEC、 I-O DATA (WN-WAPG/R) 、 PLANEX のルータの組み合わせの通信実験では一度も P2P 通信に成功しなかったことから、 Skype では Symmetric NAT 越えができないことがわかる。

6.7 実験 4 : 提案手法による NAT 越えの実験

実験 4 では、 5 章で提案した手法による NAT 越えの実験をする。

図 6.14 のようなネットワーク構成とし、 S1、 S2 グローバル IP アドレスを割り当て、 PC1 (Echo Client) および PC2 (Echo Server) にはローカル IP アドレスを割り当てた。また、 NAT a と NAT b のグローバル IP アドレスは実際の「一般家庭同士の通信」を再現するために別々のセグメントの IP アドレスを割り当てた。

さらに NAT b (iptables の時) と PC1 (Echo Client) と PC2 (Echo Server) でキャプチャーソフトの Wireshark を起動して、通信を調べた。

表 6.3 に実験で使用したマシンの仕様を示す。

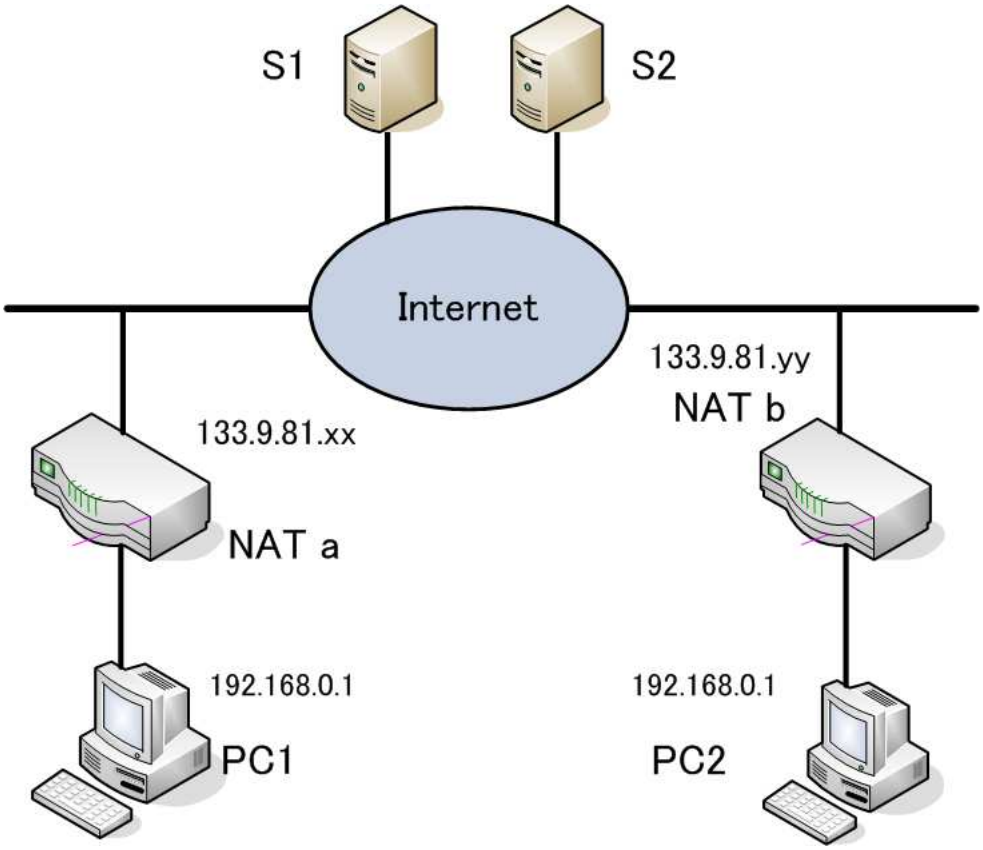


図 6.14: 実験 4 の構成

表 6.3: 実験マシンの構成

マシン名	型番	OS / IOS version
S1	DELL PowerEdge1750	FedoraCore 6
S2	DELL PowerEdge 750	FedoraCore 6
NAT b (iptables の時)	DELL PowerEdge 750	FedoraCore 5
PC1	DELL Precision	FedoraCore 5
PC1	DELL Precision	FedoraCore 5

6.8 実験 4 の結果と考察

提案手法による通信の時に NAT b (iptables の時) でキャプチャーしたデータの抜粋を図 6.15 に示す。また、 PLANEX ルータと iptables の組み合わせでの通信時に NAT b (iptables の時) でキャプチャーしたデータの抜粋を図 6.16 に示す。さらに各種 NAT との通信の結果を表 6.4 に示す。

Time	Source	Destination	Protocol	Info
0.000000	133.9.81.62	133.9.81.66	UDP	Source port: 5320 Destination port: 5320
0.000233	133.9.81.66	133.9.81.62	UDP	Source port: 5320 Destination port: 5320
0.000265	133.9.81.66	192.168.0.1	UDP	Source port: 5320 Destination port: 5320
0.000743	192.168.0.1	133.9.81.63	UDP	Source port: 5321 Destination port: 5320
0.003997	133.9.81.62	133.9.81.63	UDP	Source port: 5321 Destination port: 5320
0.004606	133.9.81.63	133.9.81.62	UDP	Source port: 5320 Destination port: 5321
0.004638	133.9.81.63	192.168.0.1	UDP	Source port: 5320 Destination port: 5321
0.004992	192.168.0.1	133.9.81.186	UDP	Source port: 5322 Destination port: 50313
0.005035	133.9.81.62	133.9.81.186	UDP	Source port: 5322 Destination port: 50313
0.005041	192.168.0.1	133.9.81.186	UDP	Source port: 5323 Destination port: 50314
0.005052	133.9.81.62	133.9.81.186	UDP	Source port: 5323 Destination port: 50314
0.012234	133.9.81.126	133.9.81.62	ICMP	Time-to-live exceeded (Time to live exceede
0.012280	133.9.81.126	192.168.0.1	ICMP	Time-to-live exceeded (Time to live exceede
0.012287	133.9.81.126	133.9.81.62	ICMP	Time-to-live exceeded (Time to live exceede
0.012295	133.9.81.126	192.168.0.1	ICMP	Time-to-live exceeded (Time to live exceede
0.012350	133.9.81.126	133.9.81.62	ICMP	Time-to-live exceeded (Time to live exceede
2.014061	133.9.81.186	133.9.81.62	UDP	Source port: 50313 Destination port: 5322
2.014108	133.9.81.186	192.168.0.1	UDP	Source port: 50313 Destination port: 5322
2.019802	192.168.0.1	133.9.81.186	UDP	Source port: 5322 Destination port: 50313
2.019831	133.9.81.62	133.9.81.186	UDP	Source port: 5322 Destination port: 50313
2.032795	133.9.81.186	133.9.81.62	UDP	Source port: 50314 Destination port: 5623

図 6.15: iptables と I-O DATA (WN-WAPG/R) の組み合わせでの通信キャプチャーデータ

Source	Destination	Protocc	Info
133.9.81.186	133.9.81.62	UDP	Source port: 5361 Destination port: 5361
133.9.81.186	133.9.81.62	UDP	Source port: 5362 Destination port: 5362
133.9.81.186	133.9.81.62	UDP	Source port: 5363 Destination port: 5363
133.9.81.186	133.9.81.62	UDP	Source port: 5364 Destination port: 5364
133.9.81.186	133.9.81.62	UDP	Source port: 5365 Destination port: 5365
133.9.81.186	133.9.81.62	UDP	Source port: 5366 Destination port: 5366
133.9.81.186	133.9.81.62	UDP	Source port: 5367 Destination port: 5367

図 6.16: iptables と PLANEX ルータの組み合わせでの通信キャプチャーデータ

表 6.4: 実験 4 の結果

	iptables	I-O	I-O2	BAFFALO	LINKSYS	PRANEX	corega	NEC	Cisco
iptables	o	o	o	o	o	o	o	o	o
I-O DATA (NP-BBRL)	/	/	o	o	o	o	o	o	o
I-O DATA (WN-WAPG/R)	/	/	/	o	o	o	o	o	o
BAFFALO	/	/	/	/	o	o	o	80%	o
LINKSYS	/	/	/	/	/	o	o	o	o
PRANEX	/	/	/	/	/	/	o	o	o
corega	/	/	/	/	/	/	/	o	o
NEC	/	/	/	/	/	/	/	/	o
cisco	/	/	/	/	/	/	/	/	/

通信の詳細

図 6.15 の通信では、 Time が 0.000000 から 0.004638 が PC2 (Echo Server) と S1 と S2 の通信の様子で、フェーズ II の F4、 F5、 F6、ならびにフェーズ III の F7 にあたる。

Time が 0.004992 から 0.005025 のところから低い TTL のパケットが順々に送り出され、フェーズ III の F8 に相当する。その後 Time が 0.012234 から 0.012350 のところからは、送り出したパケットの ICMP 時間超過メッセージが返ってくる。さらにその後パケットを送り出しながら、ICMP 時間超過メッセージが届く状態が続く。

Time が 2.014061 の時は PC1 (Echo Client) からのパケットが届き、フェーズ III の F10 に相当する。 IP アドレスとポートに問題がないので、 Time が 2.014108 の時のデータで分かるように PC2 (Echo Server) に転送されている。この後に最初に通信に成功したポート 5322 番以外のポートを Close する。

Time が 2.019802 および 2.019831 の時に、 PC2 (Echo Server) から PC1 (Echo Client) へパケットを返して、 P2P の通信が成立する。これはフェーズ III の F11 に相当する。その後 Time が 2.032795 のときに PC1 (Echo Client) から別のポート宛のパケットが届くが、すでに Close しているので、 PC2 (Echo Server) では受信しない。

ランダム型ポート変換のコントロール

図 6.16 を見ると PLANEX ルータ側が 133.9.81.186 であり、 iptables 側が 133.9.81.62 である。このときのソースポートに注目すると、ポート番号が 5361、 5362、、、と順々に増えていることがわかる。図 6.16 と図 6.5 を比べると、実験 2 の時には PLANEX ルータはランダム型のポート変換をしていた。提案手法では、 bind したポートからポート番号順にパケットを送ることによってポートを規則正しくインクリメント型のように変換させることができた。

各種組み合わせでの実験結果

表 6.4 の結果を見ると、 BAFFALO のルータと NEC のルータとの組み合わせで 80% であった以外は、全て P2P 通信に成功した。また本論文の目的である、 Symmetric NAT ルータの組み合わせに対しても全て通信に成功した。

第 7 章

結論

7.1 結論

本論文では、 2 台のサーバーとの通信による port prediction と、低い TTL に設定した UDP パケットを大量に送ることによる UDP Multi Hole Punching の方法を提案した。また提案手法を実装し、その特性について評価して、考察を行った。その結果、提案方式について以下の事柄が判明した。

- 低い TTL に設定したパケットを送り出し ICMP 時間超過メッセージが返っても NAT のポートは開いたままであった。
- ランダム型のポート変換アルゴリズムを持つルータでも bind したポートから順々にパケットを送ることによって、ポート変換をコントロールできた。
- NAT 越えの成績は Skype が 46% なのに対して提案手法は 99% 以上の確率であり、 Symmetric NAT である NEC、 I-O DATA (WN-WAPG/R) 、 PLANEX のルータの組み合わせの通信実験でも全て通信に成功した。
- STUN[1]では NAT 越えできないあるいは測定不能であった、 NEC、 I-O DATA (WN-WAPG/R)、 Cisco のルータの組み合わせの通信実験でも全て通信に成功した。

以上より提案手法のほうが既存の方法より優れていることが分かった。

[1]実験 1

表 7.1: 従来の方法との比較 (WinStun は判定結果を集計)

	WinStun	Skype	提案手法
Symmetric NAT	33%	0%	100%
全てのルータ	66%	46%	99%

7.2 今後の課題

本論文における提案方式には以下の問題が存在する。これらは今後の課題として検討する必要がある。

特殊なルータの組み合わせでの通信失敗

実験 4 (提案手法) では BAFFALO のルータと NEC のルータとの組み合わせで 5 回に 1 回は通信に失敗した。この原因を今後検証する必要がある。

リソースの使用状況

本論文の提案手法ではポートを 1000 個 bind することにしたが、これはその前の検証で Linux では 1023 個以上のポート bind するとエラーが起こるためである。そのため一瞬ではあるがポートを 1000 個開けるときにリソースが大量に使われている恐れがある。その点をさらに考察調査する必要がある。

TCP への応用

現在 UDP Hole Punching と並んで TCP Hole Punching の研究もされている。提案手法の低 TTL のパケットを大量に送ってポートを開けるアイディアを TCP のどこかに応用できるか検討することが挙げられる。

謝辞

本学士論文の作成にあたり日頃より御指導を頂いた早稲田大学理工学部の後藤滋樹教授に深く感謝致します。

また、 P2PvsNAT 班で実験環境構築および貴重なご指導を頂いた山田大輔氏、吉田傑氏に深く感謝いたします。さらに研究を進めるにあたって、貴重なアドバイスをしてくださった石井勇弥氏、河野真也氏、笹川真氏、関宏規氏、および後藤研究室の諸氏に深く感謝致します。最後に、研究室で共に助け合い、励ましあった、飯村智也氏、板倉弘明氏、岸本和之氏、木村謙一氏、下田晃弘氏、大道康徳氏、田中祐樹氏、森田慎吾氏に感謝いたします。

参考文献

[1] W.Richard Stevens, 井上尚司監訳, 橘康雄訳『詳解 TCP/IP プロトコル』ソフトバンク, 2000.

[2] 村山公保, 『基礎からわかる TCP/IP ネットワーク実験プログラミング』第 2 版, オーム社開発局 (編), 株式会社オーム社, 東京都, 2002.

[3] 竹下隆史, 村山公保, 荒井透, 苅田幸雄, 『マスタリング TCP/IP 入門編』, オーム社開発局 (編), 株式会社オーム社, 東京都, 2004.

[4] Elliotte Rusty Harold, 戸松 豊和 訳, 田和 勝 訳 , 『Java ネットワークプログラミング 第 2 版』, オライリー・ジャパン, 2001.

[5] 池嶋俊, 『入門　Skype の仕組み』, 日経 BP 社, 2005.

[6] 徳力基彦, 『図解　P2P ビジネス』, 株式会社翔泳社, 2005.

[7] 匂坂岳志 , ”状態正常化シグナリングに基づく SIP-ALG の設計と実装” , 2005.

[8] 山田大輔 , ”http トンネリングを利用した SIP セッション確立方式の検討” , 2005.

[9] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, ”STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC 3489, March 2003.
http://tools.ietf.org/html/rfc3489

[10] F. Audet, Ed., C. Jennings ”NAT Behavioral Requirements for Unicast UDP draft-ietf-behave-nat-udp-08”, October 10, 2006.
http://tools.ietf.org/wg/behave/draft-ietf-behave-nat-udp/draft-ietf-behave-nat-udp-08.txt

[11] Saikat Guha Paul Francis ”Characterization and Measurement of TCP Traversal through NATs and Firewalls”, Oct 2005.
http://nutss.gforge.cis.cornell.edu/pub/imc05-tcpnat.pdf

[12] TAKEDA, Y., "Internet draft: Symmetric NAT Traversal using STUN", June 2003. Work in progress.
http://www.cs.cornell.edu/projects/stunt/draft-takeda-symmetric-nat-traversal-0
txt

[13] Salman A. Baset and Henning Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol" September 15, 2004. http://arxiv.org/pdf/cs/0412017

[14] NAT 越えに関する技術とその仕組み 須之内雄司
http://www.skame.nu/P2Pmeeting/0409sunouchi.pdf

[15] インターネットの進化と可能性 P2P 村井 純・湧川 隆次 , 12/14/2006.
http://www.soi.wide.ad.jp/class/20060001/slides/11/35.html

[16] Tomo ' s HotLine
http://homepage3.nifty.com/toremoro/index.html

[17] Yahoo! 動画で PtoP を使ったライブ配信 –2 万 5000 人の同時接続を実現 (Cnet)
http://japan.cnet.com/news/media/story/0,2000056023,20249687,00.htm

[18] Wikipedia
http://wikipedia.org/
http://ja.wikipedia.org/wiki/Peer_to_Peer
http://ja.wikipedia.org/wiki/Gnutella
http://ja.wikipedia.org/wiki/NAT
http://en.wikipedia.org/wiki/Network_address_translation

付録 A

Winstun の実験結果

Winstun を用いた NAT の識別実験の結果を下記に示す。

iptables

```
Nat with Independend Mapping and Port Dependent Filter - VoIP will
work with STUN
Preserves port number
Does not supports hairpin of media
Public IP address: 133.9.81.186
```

IODATA (NP-BBRL)

```
Nat with Independend Mapping and Independent Filter - VoIP will
work with STUN
Does not preserve port number
Supports hairpin of media
Public IP address: 133.9.81.186
```

IODATA (WN-WAPG/R)

```
Could not reach the stun server - check server name is correct
Preserves port number
Does not supports hairpin of media
Public IP address: 0.0.0.0
```

BAFFALO

```
Nat with Independend Mapping and Port Dependent Filter - VoIP will
work with STUN
Preserves port number
Supports hairpin of media
Public IP address: 133.9.81.186
```

LINKSYS

```
Nat with Independend Mapping and Dependent Filter - VoIP will
work with STUN
Preserves port number
Does not supports hairpin of media
Public IP address: 133.9.81.186
```

PLANEX

```
Nat with Independend Mapping and Independent Filter - VoIP will
work with STUN
Preserves port number
Does not supports hairpin of media
Public IP address: 133.9.81.186
```

corega

```
Nat with Independend Mapping and Independent Filter - VoIP will
work with STUN
Preserves port number
Does not supports hairpin of media
Public IP address: 133.9.81.186
```

NEC

```
NAT Mapping is not endpoint independent - VOIP will NOT work
Does not preserve port number
Does not supports hairpin of media
Public IP address: 133.9.81.186
```

Cisco2621

```
Could not reach the stun server - check server name is correct
Preserves port number
Does not supports hairpin of media
Public IP address: 0.0.0.0
```