NAT Traversal 機能を持ったファイル送受信システムの開発

著者　　堀口日向　　指導教員　　高久有一

1. 研究背景・目的

本研究では、別々のプライベートネットワーク内にあるコンピュータ同士が、インターネット経由で、通信できるシステムを開発することを目的とする。

外部のコンピュータから、NAT 内部にあるプライベートネットワーク上のコンピュータを、直接指定して通信することはできない(便宜上、NAPT・IP マスカレードのことも、ここでは NAT と表記する)。このため、通常の方法では、別々のプライベートネットワーク内にあるコンピュータ同士が、インターネット経由で通信をすることはできないという問題がある。本研究では、この問題を解決するシステムを開発する。

2. 本システムの仕組み

本研究で開発したシステムの構成図を図 1 に示す。この図で利用している IP アドレス、ポート番号は一例である。

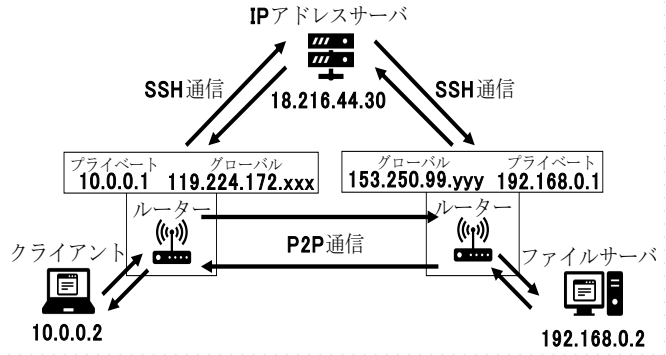

図 1．システム構成図

このシステムは、3 台のコンピュータによって構成される。まず、グローバル IP アドレスを持つサーバを用意し、これを「IP アドレスサーバ」と呼ぶ。次に、別々のプライベートネットワーク内にあるコンピュータを 2 台用意し、それぞれ、「ファイルサーバ」、「クライアント」と呼ぶ。

ここで、クライアントがファイルサーバとの TCP 接続を確立する場合、あらかじめ、ファイルサーバ、クライアントそれぞれの、TCP 接続に利用するポート番号と、ルーターのグローバル IP アドレスを、IP アドレスサーバ上のファイルに書き込んでおく。そのうえで以下の手順を実行すれば、TCP 接続が確立できる。

① ファイルサーバは IP アドレスサーバに SSH 接続し、クライアントのルーターのグローバル IP アドレスを取得して、それに SYN パケットを送る。その後、SYN パケットを送ったソケットを LISTEN 状態にする。これらを 2 分に 1 回繰り返えし、常にそのアドレスからのデータ受信を可能にする。

② クライアントは IP アドレスサーバに SSH 接続し、ファイルサーバのルーターのグローバル IP アドレスを取得し、TCP 接続を行う。

TCP 接続が確立されれば、クライアントからファイルサーバに対して、ファイルを送信するか、受信するかを指定するデータを送り、その後ファイルの内容の送受信を行う。

なお、これらの作業は、プログラムが自動で行うため、利用者はシステムの仕組みを理解する必要はない。

3. 開発・実行環境の用意

本システムでは、ファイルサーバとクライアントに、それぞれアプリケーションが必要になる。ファイルサーバ、クライアントの OS は Windows、macOS、Linux 等に幅広く対応させるため、クロスプラットフォーム開発環境である Microsoft 社の.NET Core SDK 2.1.2 にて C#で記述した。また、コマンドライン上で実行できるように、コンソールアプリケーションとして実装する。

IP アドレスサーバには、Amazon EC2(Amazon Elastic Compute Cloud)上で、Ubuntu Server 16.04 LTS 版をインストールしたマシンを用い、SSH サーバを立ち上げた。

4. システムの実装

本システムの利用者は、事前に IP アドレスサーバにユーザアカウントを作成しておく。

2 章で説明した作業を実行するための、アプリケーションを作成する。アプリケーション名は「rcpp2p」とした。また、それぞれの作業を実行するためのオプションを作成した（表 1)。

表 1. コマンドのオプション一覧

番号	オプション名	引数	動作
①	--configure	なし	設定ウィザードを開始し、ファイルサーバ名、クライアント名、使用するポート番号の設定を行う。
②	なし	なし	ファイルサーバとして、クライアントからの接続を待機する。
③	--addclient	なし	IPアドレスサーバにクライアントの情報を保存する。
④	--filelist	ファイルサーバ名	指定したファイルサーバの作業ディレクトリ内のファイル一覧を表示する。
⑤	--get	ファイルサーバ名，ファイル名	指定したファイルサーバの作業ディレクトリ内のファイルを取得する。
⑥	--put	ファイルサーバ名，ファイル名	指定したファイルサーバの作業ディレクトリ内にファイルを送信する。

ファイルサーバの場合、表 1.②オプションを実行する。そのときのアプリケーションの動作を図 2 のシーケンス図で示す。

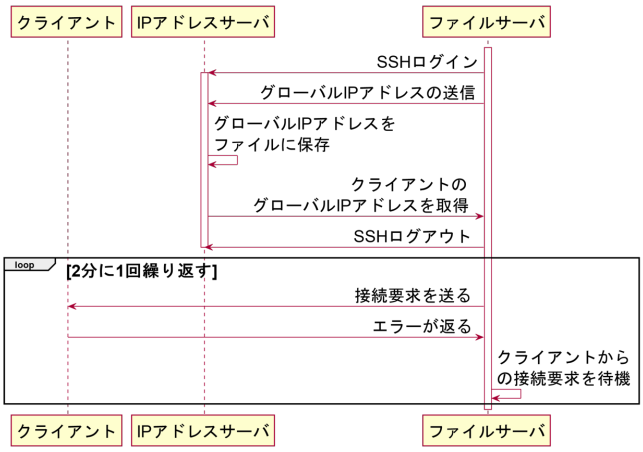

図 2.　表 1.②オプションのシーケンス図

一方、クライアントの場合、表 1.③オプションを実行した後、⑤か⑥オプションを実行する。③オプションを実行したときの動作を図 3、⑤、⑥オプションを実行したときの動作を図 4 のシーケンス図に示す。

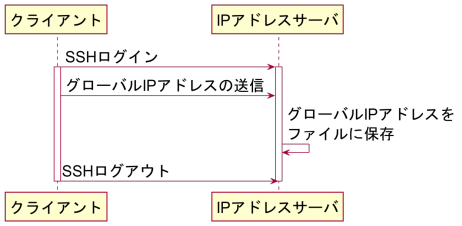

図 3.　表 1.③オプションのシーケンス図

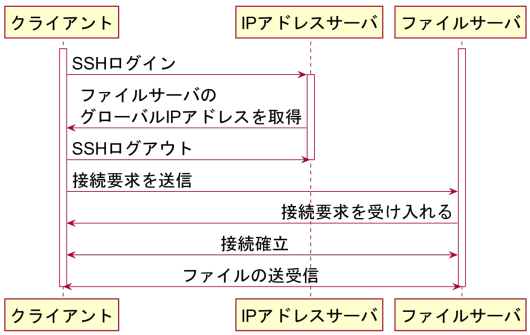

図 4.　表 1.④オプションのシーケンス図

5.　システムの動作検証

ファイルサーバのアプリケーションを実行している際の、ターミナル画面を図 2 に示す。

```
PS C:¥Users¥[redacted]> rcpp2p
UserName: test
Password:
Connecting to IP Address Server... Successful.
FileServerName: ASUS, GIPAddr: 153.159.9.[redacted], Port: 51563
Get Clients from IP Address server... Successful.
Initizlize Listener (Please wait 30 seconds) ... Successful.
Start Listener... Accepted.

Receiving and Writing photo.jpg (991.82KB)
100% [====================] 991.82KB 276.35KB/s Complete.
```

図 2.　ファイルサーバアプリの実行結果

コマンド実行後に、IP アドレスサーバにログインするためのユーザ名とパスワードを入力すると、クライアントの情報を IP アドレスサーバより取得したのち、クライアントからの接続要求を待機する。次節で説明するクライアントのアプリケーションからの接続が来ると、それを受け入れて接続する。そして、ファイルの受信をしている。

クライアントからファイルサーバに対して「photo.jpg」ファイルを送信するときのコマンドを実行した結果を図 3 に示す。また実行したコマンドを以下に示す。

```
rcpp2p --put ASUS photo.jpg
```

```
PS C:¥Users¥[redacted]> rcpp2p --put ASUS photo.jpg
UserName: test
Password:
Connecting to IP Address Server... Successful.
ClientName: Desktop, GIPAddr: 119.224.174.[redacted], Port: 63423
Get FileServer from IP Address server... Successful.
Connecting to FileServer (FileServerName: ASUS, IPAddr: 153.159.9.[redacted]
Successful

Sending photo.jpg (991.82KB)
100% [====================] 991.82KB 325.08KB/s Complete.
```

図 3.　クライアントアプリの実行結果

6.　まとめ・今後の課題

本研究では、別々のプライベートネットワーク内にあるコンピュータ同士が、インターネット経由で、通信できるシステムを考案し、それを利用したファイル送受信ソフトウェアを開発した。

本システムを用いれば、例えば、5 分間の HD 動画（1GB）なら、約 10 分で送ることができる。これらを、Dropbox 等のクラウドストレージでやるとすれば、最低限アップロードとダウンロードの時間があるため、それぞれ倍以上の時間がかかることになる。

本システムは、ファイルの送受信を含むデータのやり取りに暗号化をしていない。そこで情報漏洩を防ぐため、ハイブリッド暗号方式を用いて、やり取りするデータの暗号化を行いたい。また、本システムを応用すれば、ファイルの送受信だけでなく、リモートシェルや高度なファイル転送（ファイル転送の中断・再開など）も可能である。つまり、OpenSSH でサポートされている ssh, sftp コマンドを、別々のプライベートネットワーク内にあるコンピュータ同士で行うことができるようになる。なので、今後はこのようなアプリケーションの開発も行いたい。

7.　参考文献

mat2uken（2016）
「NAT Traversal って知ってますか」，
<https://tech-blog.cerevo.com/adventcalendar2016/advent24/> 2017/12/18 アクセス