統合的なVPNサーバソフトウェアに関する研究

2017年3月

登　大遊

統合的なVPNサーバソフトウェアに関する研究

登　大遊

システム情報工学研究科
筑波大学

2017年3月

概要

インターネット上に安全なオーバーレイネットワーク構築する手法である仮想プライベートネットワーク (VPN) には、多様なプロトコルや実装がある。従来手法では、多様な VPN プロトコルを使用するクライアントに対してサービスを提供する場合、単一、または少数の VPN プロトコルを実装した VPN サーバソフトウェアを複数組み合わせる必要があり、管理が難しい。システム管理者は、異なる複数の VPN サーバソフトウェアの特性を理解し、相互依存関係や周辺システムとの相性、相互干渉の問題などに対応しなければならない。VPN サーバを検閲回避目的で設置するボランティアにとっても、多数のユーザが多様なクライアントで利用できる VPN サーバを設置する場合のインストールと管理が難しい。本研究では、これらの問題を解決するため、単一のインスタンスで、多様な VPN プロトコルを統一的に扱うことができるようにし、VPN サーバの設定や管理を簡単にするための、統合的な VPN サーバソフトウェアを実現する。

統合的な VPN サーバでは、レイヤ 2 の VPN プロトコルと、レイヤ 3 の VPN プロトコルの両方をサポートしなければならない。このレイヤの違いの解決のために、本研究では L2 アダプタを提案し、L2 と L3 とを統一的に扱えるようにする。

統合的な VPN サーバは、複数の VPN プロトコル間で、ユーザ認証、セキュリティ設定の適用、ログの管理、IP アドレス割当管理などの処理を単一化することにより、管理を簡単にする。本研究では、VPN サーバ内の通信処理を、すべて L2 に統一することで、これらの処理を単一化できるようにする。

本研究では、提案手法に基づいて、統合的な VPN サーバソフトウェアである SoftEther VPN Server を実装する。SoftEther VPN Server は、L2TP/IPsec、SSTP、OpenVPN (L2 および L3)、L2TPv3/IPsec、EtherIP/IPsec および SoftEther VPN Protocol (SEVP) を 1 つの VPN サーバインスタンスでサポートする。本ソフトウェアは、さらに、異なる VPN プロトコル間を接続した場合において、従来方式と比較して、最大 7% 程度のスループットの向上を実現した。

本研究では、実装した統合的 VPN サーバソフトウェアを用いて、政府によるインターネット検閲を回避するための、ボランティアベースの分散型公開 VPN 中継システムである VPN Gate を実現する。政府の検閲当局は、このような公開 VPN 中継システムの中継サーバの IP アドレスを、公開されているサーバリストから発見し、遮断する。本研究では、新たに、無実の IP 混入手法を提案し、検閲当局にサーバリスト内のすべての中継サーバが本物であるかどうかの検証を強いる。本研究では、さらに、当局による検証活動を防ぐため、新たに協調的スパイ発見手法を実装し、検閲当局の検証用 IP アドレスを発見できるようにする。これらの手法により、VPN Gate では、政府による検閲用ファイアウォールによる遮断に対する耐性を実現する。

目次

第 1 章

序論

第 1 章では、本研究の背景、目的、および提案手法の概略について述べる。

1.1　研究の背景と問題

インターネット上に安全なオーバーレイネットワークを構築する方法として、仮想プライベートネットワーク (Virtual Private Network: VPN) がある。現在、幅広く使われている VPN システムでは、VPN サーバと VPN クライアントとが通信を行う。VPN サーバと VPN クライアントとの間で利用される VPN プロトコルには、多様な種類がある。VPN サーバ、VPN クライアントとも、多様な実装がある。主な VPN の利用目的は、企業や学校などの組織における業務や、個人による政府のインターネット検閲の回避である。

企業や学校などの組織における VPN の利用方法は、2 種類に分けることができる。1 種類目の利用方法は、リモートアクセスである。リモートアクセス型の VPN を利用すれば、たとえば会社の Local Area Network (LAN) に、自宅や外出先などのコンピュータを、物理的には離れているにもかかわらず、仮想的に直接接続している状態とすることができる。自宅や外出先などのコンピュータの OS には、Windows、Mac OS X、Linux などの複数の種類がある。また、スマートフォンなどの携帯型コンピュータの OS にも、iOS や Android などの複数の種類がある。これらの OS 上からは、標準搭載の VPN クライアントソフトウェア、あるいはユーザがインストールする VPN クライアントソフトウェアを用いて、VPN サーバに接続することができる。リモートアクセスのための VPN プロトコルは複数存在する。OS ごとに利用できる VPN クライアントの実装が異なり、それぞれの VPN クライアントで利用できる VPN プロトコルが異なる。たとえば、Windows では Layer 2 Tunneling Protocol (L2TP)/IPsec、Secure Socket Tunneling Protocol (SSTP)

および SoftEther VPN が利用できるが、Mac OS X ではこれらの中では L2TP/IPsec しか利用できない。Linux ではディストリビューションによって簡単に利用可能な VPN プロトコルが異なる。例えば CentOS では OpenVPN を簡単に利用できるが、L2TP/IPsec を利用することは難しい。

企業や学校などの組織における 2 種類目の VPN の利用方法は、拠点間接続である。拠点間接続型の VPN を利用すれば、たとえば本社の LAN に 1 つ以上の支店の LAN を接続し、すべての支店の LAN と本社の LAN とを、仮想的に 1 つの LAN とすることができる。それぞれの拠点にあるコンピュータ同士は、特別な設定なしに、また VPN クライアントのインストールなしに、相互に通信することができるようになる。リモートアクセスの VPN プロトコルと同様に、拠点間接続型の VPN プロトコルにも、複数の種類が存在し、ネットワーク装置やサーバの機種や OS によって利用可能な VPN プロトコルが異なる。例えば、Windows や Linux を拠点間接続用 VPN 装置として利用する場合は、OpenVPN が使用できるが、Cisco 社の VPN 対応ルータでは OpenVPN がサポートされていない。Cisco 社の VPN ルータは、L2TPv3 over IPsec をサポートしている一方、同等の目的のための全く異なるプロトコルである EtherIP over IPsec をサポートしている NEC 社の VPN 対応ルータは、L2TPv3 over IPsec をサポートしていない、というような状況がみられる。

個人における VPN の利用目的の 1 つに、政府によるインターネット検閲の回避がある。政府がインターネット検閲を実施している国は、複数存在する。これらの国には、インターネット上の特定のサイトへのアクセスや、特定のキーワードを用いた検索の実施に関する通信を、自動的に遮断するための検閲用ファイアウォールが設置されている。このような国の内側から、規制されたインターネット上のサイトにアクセスしたいという需要が存在する。また、政府がこのような検閲を実施していない場合でも、政府により、通信内容が盗聴し記録されている可能性がある。これらの政府による検閲やその他の傍受システムにおいては、ユーザの接続元 IP アドレスと、ユーザの通信内容とが関連付けて保存される場合が多い。このような、自己の通信記録が政府によって収集、保存されないよう望むユーザは多い。そのようなユーザは、検閲の回避や匿名性の確保のために、しばしば VPN を利用する。VPN の通信内容は暗号化されるため、検閲用ファイアウォールや傍受システムは、アクセス先や通信の内容を検査することができなくなる。これにより、検閲や傍受を回避することができる。

リモートアクセスや拠点間接続において、ネットワーク管理者は、従来、サポートしたい VPN プロトコルすべてをカバーするため、各 VPN プロトコルごとに、1 つずつ VPN

サーバソフトウェアまたはVPNルータなどの専用機器をインストールし、設定および管理をしなければならない。汎用的なサーバコンピュータを用いる場合、1台または複数台のサーバコンピュータ上で、複数の種類のVPNサーバプログラムを同時に起動する必要がある。この場合、それぞれのVPNサーバプログラムごとに、認証やアクセス制御などの設定や、IPアドレスの割り当て設定など、同一の目的の設定を個別に行う必要があり、管理が難しくなるという問題がある。また、異なるVPNプロトコルのクライアント同士で通信が発生すると、複数のVPNサーバソフトウェア間で通信が発生するため、オーバーヘッドが発生し、パフォーマンスが低下するという問題がある。システム管理者は、異なる複数のVPNサーバソフトウェアの特性を理解し、相互の依存関係や周辺システムとの相性、同時に併用をする場合に発生する相互干渉の問題などについて配慮をしつつ、VPNシステムを設計する必要がある。

さらに、VPNサーバソフトウェアは、特定のOSに依存している場合が多く、その場合、特定のOSにしかインストールできない。複数のVPNサーバソフトウェアの相互運用が必要になる場合で、かつ、物理的には1台のサーバコンピュータにまとめたい場合は、仮想マシン (Virtual Machine: VM) や仮想スイッチを用いることになる。この場合、VMや仮想スイッチの使用方法に関する知識や、これらの特性に起因するトラブルシューティングが必要となる。このような複雑なサーバ環境でVPNサーバを組み合わせて実行すると、しばしば、パフォーマンスの低下が発生する。

従来のVPNサーバソフトウェアの多くは、OSのネットワーク機能の一部として実装されていることが多い。また、OSとは別に提供されるVPNサーバソフトウェアであっても、ユーザモードで動作するプログラムだけでなく、OSの既存のカーネル内のモジュールを利用したり、新たにカーネルモジュールのインストールを必要とするものが多い。

図1.1に、従来のVPNサーバの問題を示す。従来のVPNサーバソフトウェアは、多くの場合、自らの実行時に、VPNプロトコルごとに、個別のユーザモードのプログラムを起動して実行する。図1.1では、L2TPサーバ、OpenVPN (L3) サーバおよびEtherIPサーバの3種類のVPNサーバソフトウェアを実行している。これらのサーバは、異なるプロセスとして実行される。VPNサーバの管理者は、複数のVPNプロトコルごとに、複数のVPNサーバに対して管理設定を繰り返す必要がある。また、それぞれのVPNサーバからのログも個別に出力される。

図1.1では、L2TPサーバとOpenVPNサーバは、L3のVPNプロトコルを処理する。別のVPNプロトコルや、外部の物理的なLANとの間でのIPパケットの交換のために、OSの有するIPルーティング機能を呼び出す。EtherIPサーバは、L2のVPNプロトコル

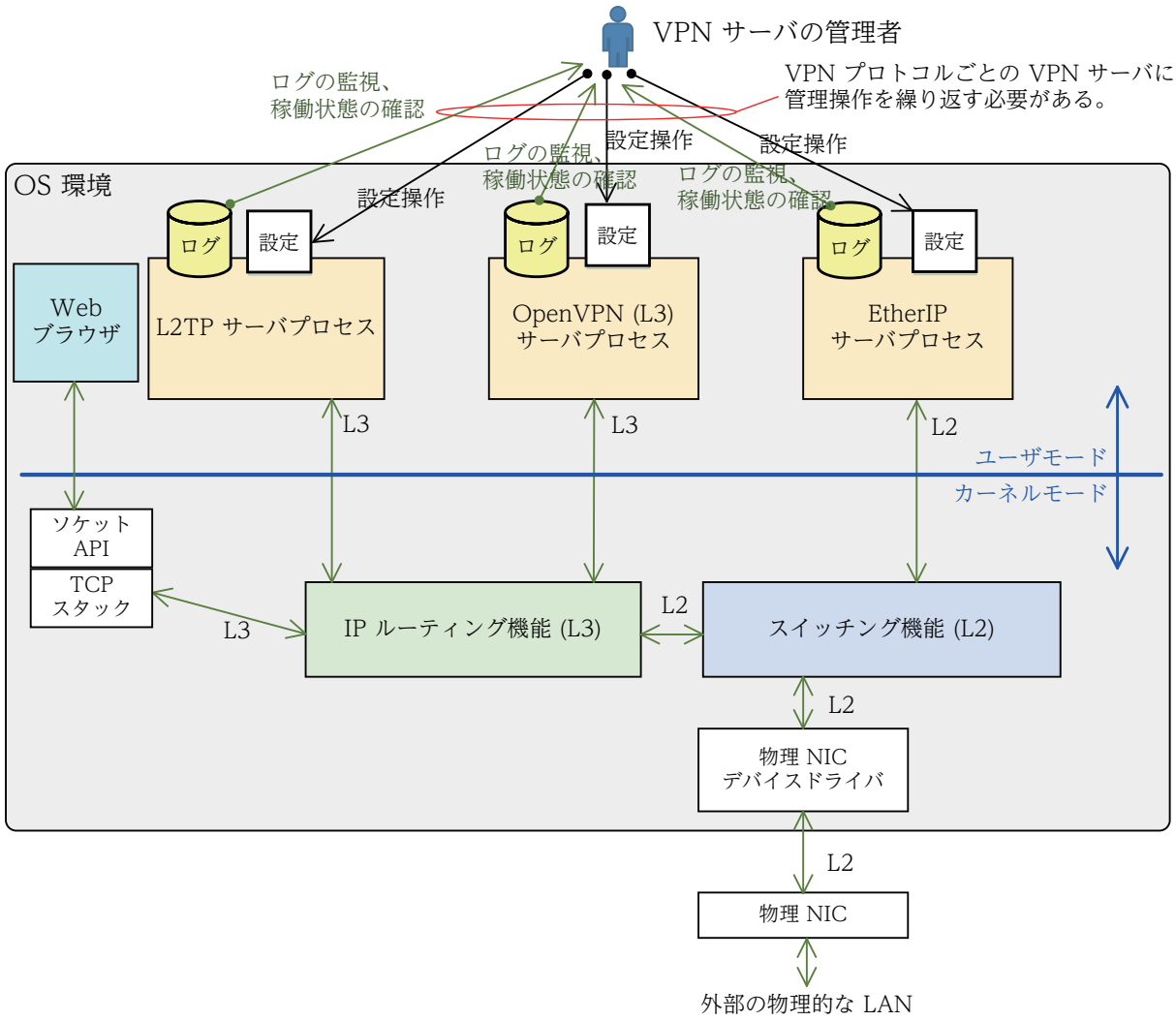


図 1.1: 従来の VPN サーバの問題

を処理する。別の VPN プロトコルや、外部の物理的な LAN との間での MAC フレームの交換のために、OS の有するスイッチング機能を呼び出す。管理者が複数の VPN プロトコルをサポートする VPN サーバを構築する際には、OS のルーティング機能やスイッチング機能に依存するため、これらの機能間を接続する設定が必要になる。また、VPN サーバソフトウェア以外のプログラムにも大きな影響が生じる。複数の VPN サーバをインストール、設定するときに、OS のスイッチング機能やルーティング機能の設定に誤りがあると、VPN サーバとは無関係の Web ブラウザなどのプログラムも通信できなくなる。

また、図 1.1 では、VPN サーバ間の IP パケットの交換と、Web ブラウザの使用する TCP/IP パケットの送信に、単一の OS のルーティング機能が共有される。VPN サーバ間に限定した IP サブネットを運用する場合において、その IP サブネットと重複する IP

アドレスを持つ外部の物理的な LAN のホストがある場合を想定する。この場合、Web ブラウザが外部の物理的な LAN のホストにアクセスしようとしても、VPN サーバに向かってルーティングがされてしまい、Web ブラウザは、正しく宛先ホストと通信できなくなる。

さらに、L2TP/IPsec プロトコルの VPN サーバは、PPP プログラムと、IPsec プログラムを実行する必要がある。加えて、VPN サーバを流れる通信にパケットフィルタを適用する場合は、パケットフィルタプログラムを起動する必要もある。複数の連携する必要があるプログラムを、事前にインストールをしなければならない。

検閲回避のための VPN システムを実現する場合も、リモートアクセスと同様に、VPN サーバの問題に直面する。強力でスケーラブルな検閲回避システムを構築するために、ボランティアを組織化し、多数の VPN サーバを動作させる方式がある。中継用 VPN サーバの台数が増加すると、多数の IP アドレスで中継サーバが動作することになるため、検閲の回避に有利となる。また、1 台あたりの負荷が減少し、パフォーマンスが向上する。したがって、できるだけ多くのボランティアに参加をしてもらったほうが良い。そのためには、ボランティアの日常的な PC の利用に影響が出ることなく、VPN 中継サーバを稼働させることができることが要求される。できるだけ多くの VPN ユーザをサポートするためには、多様な VPN クライアントソフトウェアからの、多様な VPN プロトコルを、同時にサポートすることが望ましい。これらを実現することは、従来の VPN サーバソフトウェアでは難しい。

検閲回避のための VPN システムにおける VPN サーバは、検閲をしたい政府によって発見された場合、政府のファイアウォールによって通信が遮断されてしまう。中継用 VPN サーバの IP アドレスのリストを、ユーザに対して、インターネット上の Web サイトや SNS などで公開すると、この IP アドレスのリストは、検閲当局も入手できてしまう。

1.2 本研究の目的と提案手法

本研究の目的は、このような VPN サーバに関する問題を解決することである。本研究では、統合的な VPN サーバソフトウェアにより解決することを提案する。統合的な VPN サーバソフトウェアとは、管理者が多様な VPN プロトコルを、差異を意識する必要なく同時に利用でき、また、他の特定の OS やライブラリへの依存度が低く、1 つのインスタンスとして動作するような VPN サーバソフトウェアである。

図 1.2 に、統合的な VPN サーバによる解決手法を示す。統合的な VPN サーバソフトウェアは、1 つのプロセス内に、複数の VPN プロトコルを処理するモジュールを有する。

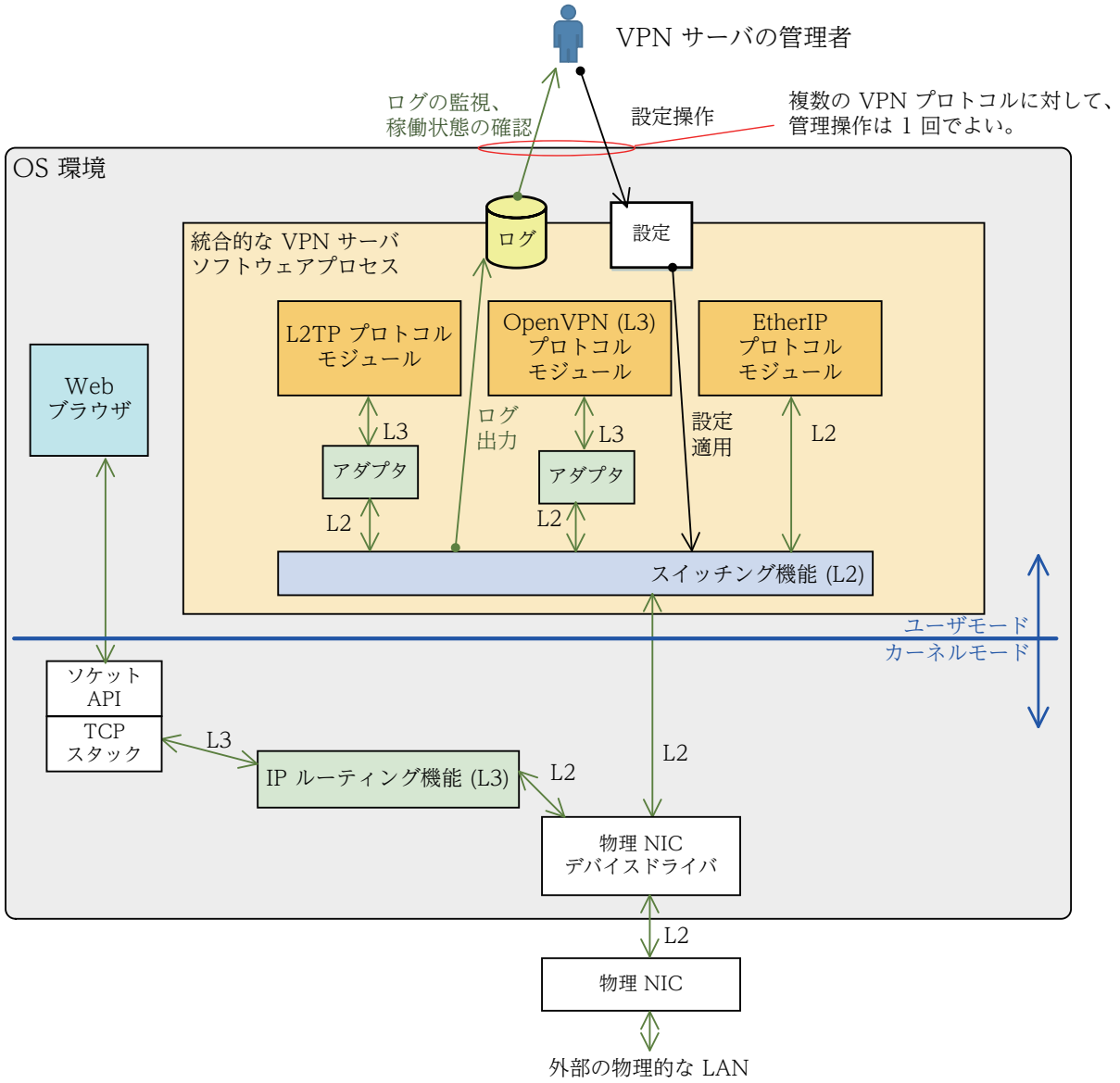


図 1.2: 統合的な VPN サーバによる解決手法

管理者は、統合的な VPN サーバソフトウェアを 1 つインストールするだけで、多様な VPN プロトコルをサポートし、様々な VPN クライアントソフトウェアからの VPN 接続を実現することができる。スイッチング機能も、プロセス内に実装される。スイッチング機能は、複数のレイヤ 2 の VPN サーバ間のメッセージ交換を実現する。また、L3 VPN サーバとスイッチング機能との間では、アダプタにより、レイヤ 3 とレイヤ 2 とを変換する。管理者は、VPN サーバソフトウェアとは別に、OS のスイッチング機能やルーティング機能を設定する必要がなくなる。また、統合的な VPN サーバソフトウェアでは、ユー

ザ認証、セキュリティ設定の適用、ログの管理、IP アドレスの割り当て管理などの管理業務を行う際に、1 回の操作だけで、すべての VPN プロトコルの通信にその設定が適用される。各 VPN サーバモジュールのログも、1 つに統一されて出力される。さらに、VPN サーバとは無関係の Web ブラウザなどのアプリケーションの通信に使用される TCP/IP スタックと、統合的な VPN サーバとは、OS の IP ルーティング機能を共有しない。そのため、Web ブラウザなどのアプリケーションは、VPN サーバをインストール、設定する際に通信ができなくなるといった影響を受けない。

管理者は、VPN サーバソフトウェアのために複数の OS 環境や複数台のサーバコンピュータ、複数機種の VPN ルータを用意する必要はない。管理者は、すでに有している OS やサーバコンピュータに、統合的な VPN サーバソフトウェアをインストールできる。これにより、複数 VPN サーバソフトウェアを併用する場合の相互干渉の問題や周辺システムとの相性に起因する難しい問題に直面することなく、簡単に、複数の VPN プロトコルをサポートすることができる。

統合的な VPN サーバソフトウェアは、検閲回避のための VPN システムにおける従来の問題も解決する。ボランティアは、コンピュータに気軽に統合的な VPN サーバソフトウェアをダウンロードし、一般ユーザ権限で稼働させることができる。この際には、ボランティアの日常的な PC の利用に影響が出ることがない。

加えて、本研究では、統合的な VPN サーバソフトウェアを用いて強力でスケーラブルな検閲回避システムを実現する。そのため、政府当局による中継 VPN システムの検出と遮断に対する耐性を実現する。

上記の提案手法の有効性を示すため、本研究では、以下の 2 つの VPN ソフトウェアを設計および実装する。

1. マルチプロトコル対応のクロスプラットフォームな VPN サーバソフトウェア SoftEther VPN Server。
2. 検閲用ファイアウォールを回避するための、ボランティアを組織化する分散型公開 VPN 中継システム VPN Gate。

統合的な VPN サーバの実現においては、様々な問題が生じる。本研究では、これらの問題を、新たな手法で解決する。

統合的な VPN サーバにおいて、複数の VPN プロトコルを統一的にサポートするためには、VPN プロトコル間の差異を吸収する必要がある。特に、レイヤ 2 の VPN プロトコルと、レイヤ 3 の VPN プロトコルとを統一的に扱うことは、難しい。本研究では、こ

の問題を解決するために、L2 アダプタという新しい手法を提案する。

統合的な VPN サーバにおいては、システム管理者が VPN プロトコルの違いにかかわらず、パケットフィルタやセキュリティポリシーなどのセキュリティ設定や、ユーザ管理、ログ管理などの管理運用などを統一的に扱う必要がある。本研究では、すべての VPN プロトコルを、共通のバスとして実装する単一の L2 スイッチを経由させ、その L2 スイッチ上でセキュリティや管理モジュールを実装する手法を提案する。

本研究では、統合的な VPN サーバを用いて、検閲回避システムを構築を実現する。そのために、ボランティアを組織化する。この VPN 中継システムにおいて、当局による検閲遮断に対する耐性を実現するため、本研究では、無実の IP アドレス混入手法および協調的スパイ発見手法を提案する。

1.3 SoftEther VPN Server

統合的な VPN サーバは、多数の種類の VPN プロトコルに対応しなければならない。これらの VPN プロトコルには、互いに差異がある。最も大きな差異は、レイヤの違いである。主要な VPN プロトコルは、カプセル化の対象として、レイヤ 2 (Ethernet フレーム) を対象とするものと、レイヤ 3 (IP パケット) を対象とするものの 2 種類に大別することができる。レイヤ 2 (Ethernet フレーム) のメッセージを交換する動作は、スイッチングと呼ばれ、レイヤ 2 スイッチの内部の処理と同等である。一方、レイヤ 3 (IP パケット) のメッセージを交換する動作は、ルーティングと呼ばれ、レイヤ 3 ルータの内部の処理と同等である。ネットワークハードウェアの例をみると、レイヤ 2 のスイッチとレイヤ 3 のルータは、大きく異なる回路やファームウェアで実現されている。統合的な VPN サーバの内部においては、同様に、レイヤ 2 の VPN プロトコルと、レイヤ 3 の VPN プロトコルとを統一的に扱うために、大きく異なる 2 種類のメッセージ交換方式が必要となる。すなわち、統合的な VPN サーバが、この大きく異なる 2 種類のレイヤ間を、シームレスに接続しなければならない。従来の VPN サーバでは、単一の VPN サーバはレイヤ 2 またはレイヤ 3 のいずれかの VPN プロトコルしか取り扱っていなかったため、この問題を解決する必要はなかった。

本研究では、レイヤ 2 とレイヤ 3 とをシームレスに接続する方法として、L2 アダプタを提案する。L2 アダプタとは、レイヤ 3 方式の VPN の通信パケットに対して動作する変換モジュールである。VPN プロトコル間のメッセージの交換は、いずれのレイヤの VPN プロトコルであっても、レイヤ 2 の Etheret フレームとして統一して取り扱う。レイヤ 3 方式の VPN プロトコルは、L2 アダプタにより、一旦レイヤ 2 方式に変換される。

本研究で実装する統合的な VPN サーバソフトウェアである SoftEther VPN Server は、L2 アダプタにより、レイヤ 3 方式の VPN プロトコルを透過的にレイヤ 2 に変換する。さらに、各 VPN プロトコルごとにプロトコルモジュールを実装し、仮想レイヤ 2 スイッチと接続する。これにより、SoftEther VPN Server は、VPN プロトコル間の差異を吸収し、相互接続を可能にする。

本研究では、複数の VPN プロトコル間のメッセージを交換する共通のバスとして、L2 スイッチを用意し、そこで、セキュリティ設定や、ユーザ管理、ログの保存などを行う。L2 アダプタにより、すべての VPN プロトコルのメッセージはレイヤ 2 に変換される。この手法で実現される VPN サーバでは、変換後のレイヤ 2 のメッセージである Ethernet フレームに対して、パケットフィルタやセキュリティポリシーなどを適用する。

本研究では、これらの提案手法を用いて実装した統合的 VPN サーバソフトウェアである SoftEther VPN Server の評価を行った。管理者は、各 VPN プロトコルの差異を意識せずに、設定・運用ができるようになった。異なる VPN プロトコル間の通信速度について、従来方式と比較して、最大 7% 程度の性能向上を実現した。ユーザ管理やネットワーク機能を仮想化するマルチテナント機能、および複数の OS への移植性も実現した。この SoftEther VPN Server を、2014 年 1 月にオープンソースソフトウェアとして公開した。SoftEther VPN Server は、2016 年 12 月 20 日までに世界中で 1,009,000 回インストールされた実績を有する。本論文では、これらの結果に基づき、統合的な VPN サーバソフトウェアにより VPN サーバの管理に関する問題を解決できることを示す。

1.4 VPN Gate

検閲回避システムを実現するために、本研究では、まず、中継 VPN サーバの数を大量に用意する。このために、世界中からボランティアを募り、ボランティアの PC 上で中継 VPN サーバを立ち上げてもらう。そして、そのサーバの IP アドレスのリストを公開する。ユーザは、いずれかのサーバに接続できれば VPN 接続が可能である。一方、検閲政府の側は、すべてのサーバを遮断しなければならなくなる。政府による検閲用ファイアウォールの容量をオーバーフローさせることをねらう。

さらに、本研究では、中継 VPN サーバの IP アドレスのリストに無実の IP アドレスを混入する手法を提案する。無実の IP アドレスとは、VPN 中継サーバとは全く無関係の IP アドレスで、かつ、その IP アドレス宛の通信を遮断をされると、国内から検閲当局に対して苦情が多数寄せられるおそれがあるような、重要なサーバ (たとえば DNS ルートサーバなど) の IP アドレスである。無実の IP アドレスは、常時混入する必要はなく、不

定期に、少量を混入するだけで良い。不定期に無実の IP アドレスを混入し、検閲当局に対してそのような無実の IP アドレスが時々混入されている可能性に気付いてもらえれば、検閲当局は、それらの IP アドレスをそのままファイアウォールの遮断リストに投入することはしなくなる。

しかし、無実の IP アドレスの混入手法を実装したとしても、検閲当局は、サーバリスト上のすべての IP アドレスに対して、VPN 接続を試行することにより、その IP アドレスの真正検証を行うことができる。検閲当局は、真正検証に失敗した IP アドレスを、検閲用ファイアウォールへの投入対象から除外することにより、無実の IP アドレス混入手法に対する対策を実現できてしまう。

この問題を解決するため、本研究では、新たな手法として、協調的なスパイ発見手法を提案する。協調的なスパイ発見手法とは、すべてのボランティアの VPN 中継サーバが協調動作することにより、特定の IP アドレスが無実の IP アドレスであるか否かを識別するための VPN 接続を試行する検閲当局のコンピュータ (スパイコンピュータ) であると疑われる IP アドレスのリスト (スパイリスト) を生成し、共有することにより、スパイコンピュータを検出する手法である。

これにより、各 VPN 中継サーバは、スパイリストに含まれる IP アドレスからのパケットを無視するようになる。その結果、検閲当局は、サーバリストに含まれるある IP アドレスが、本物の VPN 中継サーバであるか、または混入された無実の IP アドレスであるかを見分けることが困難になる。

本研究では、SoftEther VPN Server に、この検閲回避の仕組みを組み込んだ VPN Gate システムを実装し、公開した。そして、中国政府が運用する Great Firewall (GFW) of China に対する検閲耐性の評価を行った。本研究の手法である、無実の IP アドレスの混入手法および協調的スパイ検出手法を用いない場合、GFW は 81% の VPN サーバの遮断に成功した。しかし、これらの提案手法を用いた場合、遮断率を 25% に低下できた。本システムは、2016 年 7 月時点で、1 日あたり中国の約 34,000 個のユニーク IP アドレス、および全世界からの約 234,000 個のユニーク IP アドレスから利用されており、全世界で平均約 6.7 Gbps のトラフィックを分散処理している。

1.5　本論文の構成

第 2 章では、本研究の関連研究について述べる。また、VPN を実現するための手法やプロトコルについて、それぞれの特徴や問題点を述べる。

第 3 章では、統合的な VPN サーバソフトウェアを実現する手法を提案し、それらの手

法を用いた統合的な VPN サーバソフトウェアである SoftEther VPN Server の設計、実装および評価について述べる。また、VPN プロトコル間のセキュリティ設定や運用管理の統一化手法等の実装および評価について述べる。

第 4 章では、統合的な VPN サーバソフトウェアにより構築される検閲回避システムである VPN Gate の設計および実装について述べる。

第 5 章は、本研究の結論をまとめ、今後の研究の展開を述べる。

第 2 章

関連研究

第 1 章では、本研究の背景、目的および構成について述べた。第 2 章では、本研究に関係する従来手法について述べる。

2.1 VPN サーバの設計や実装

2.1.1 Click モジュール型ルータ

ルータやその他のネットワークデバイスを構築するためのソフトウェアアーキテクチャとして、Click モジュール型ルータ [73] がある。Click モジュール型ルータは、IP ルーティングに必要な各種処理を分解して Elements と呼ばれる各モジュールに実装し、それらのモジュール間におけるパケットの流れやステートの共有などのワークフローをテキスト言語や Graphical User Interface (GUI) で記述することが可能なルータの設計手法に関する研究である。通常モノリシックなプログラムとしてモジュール間が密に結合され実装されている IP ルータと異なり、Click ルータのモジュール間の結合は疎である。新しくルータを実装したいと考えるユーザは、モジュール間を結合し、希望する設計のルータを作成することができる。Click ルータのルーティング速度は、同等の処理を通常の Linux カーネルのルーティング速度と比較してちょうど 10% 程度の速度低下が見られる程度であると報告されている [73]。

多くの開発者が Click を研究に利用している。たとえば、SMP Click [10] は、Click を用いて、マルチプロセッサ環境で高速に動作するソフトウェアルータを実装した例である。SMP Click の研究では、マルチプロセッサ用に最適化されたバッファやキューの管理手法を提案している。これにより、複数のプロセッサに対してアダプティブにパケットが分散され、ルーティング処理が並列的に実行される。これにより、プロセッサ数が 1 個

の場合と比較すると、4 個の場合は、約 3.8 倍の高速化を実現している。

また、特定のネットワークに特化したルータを実装する研究でも、Click が使用されている [56]。この研究では、Wireless Personal Area Network (WPAN) [30] という短距離通信に特化した無線プロトコルによるネットワークと、IP ネットワークとの間のメッセージ交換を行うためのエッジルータを、Click を用いて実装している。この研究で実装されたルータの管理者は、設定変更や挙動の変更を、ルータを再起動することなく動的に実施することができる。たとえば、ルータのデータプレーンと呼ばれるパケットの処理部分と、ルーティングプロトコルとの間を、動的にインタラクションさせることができる。

大規模なネットワークをシミュレーションする研究でも、Click が使用されている。たとえば、Hydra は Click を使用して、マルチホップの無線ネットワークをシミュレーションすることができるテストベッドである [59]。無線ネットワークのシミュレーションを行うためには、たとえば Orthogonal Frequency-Division Multiplexing (OFDM) [75] や Multiple-Input and Multiple-Output (MIMO) [82] などの無線特有の機能をサポートした、特殊な Media Access Control (MAC) 副層 [54] および物理層のモジュールが必要である。Hydra では、Click におけるモジュールとしてこれらの無線モジュールを実装することにより、無線ネットワークのシミュレーションシステムを実現している。

VPN サーバを実装する場合において、Click のようなモジュール型実装手法が有効であるかどうかは未知である。本研究で実装する統合的な VPN サーバソフトウェアにおいては、Click モジュール型ルータと同様に、パケットの処理の単位ごとにモジュールを作成してそのモジュール間でパケットの受渡しを行うことにより、複数のプロトコルが組み合わさっている複雑な VPN プロトコルを処理できるようにする。ただし、Click モジュール型ルータではユーザによるモジュール間の動的な結合や初期化パラメータを指定するための仕組みがあり、また各モジュール間における CPU のスケジューリングも独自に実装しているが、本研究の実装においてはモジュール間の結合はコンパイル時に定まる静的なものとし、CPU スケジューリングは OS の機能を利用する。

2.1.2 OpenVPN、poptop および openl2tpd

オープンソースのフリーソフトウェアとしてリリースされている VPN サーバソフトウェアがある。

OpenVPN はオープンソース VPN ソフトウェアである [26]。OpenVPN は独自の OpenVPN L3 および L2 プロトコルを用いて通信を行う。OpenVPN サーバソフトウェアは、OpenVPN プロトコルにしか対応しておらず、L2TP/IPsec、SSTP、EtherIP な

どの多様な VPN プロトコルはサポートしていない。そのため、OpenVPN サーバソフトウェアに VPN 接続をしようとするユーザは、OpenVPN のクライアントソフトウェアのインストールを必要とする。また、OpenVPN サーバソフトウェアの 1 つのインスタンスは、OpenVPN の L3 または L2 のいずれかの VPN プロトコルのみをサポートする。L3 または L2 のいずれの VPN プロトコルをサポートするかは、起動時に読み込まれる設定ファイルで指定する。1 つの OpenVPN サーバインスタンスで、L3 および L2 の両方の VPN プロトコルを同時にサポートすることはできない。システム管理者は、これらを同時にサポートし、VPN プロトコル間の通信を実現したい場合は、2 つのインスタンスを起動し、OS の有するルーティング機能、スイッチング機能および仮想ネットワークインターフェイス機能を用いて、L2 と L3 との間のメッセージ交換を実現する必要がある。この場合、OS の種類やバージョンごとに固有の設定が必要となり、設定や管理が難しい。また、同一の OS 上で動作する他のネットワークアプリケーションの通信にも影響が生じる可能性がある。さらに、L3 および L2 の 2 つの OpenVPN サーバインスタンスを実行している場合は、設定ファイルが別々であるので、システム管理者は、初期設定を行ったり、設定を変更したりする際に、同一の操作を 2 回行う必要がある。

Poptop [35] および openl2tpd [44] も、オープンソース VPN サーバソフトウェアである。Poptop は Point-to-Point Tunneling Protocol (PPTP) のサーバ機能を有する。openl2tpd は L2TP のサーバ機能を有する。いずれの VPN サーバソフトウェアも、単一の VPN プロトコルにしか対応していない。これらの VPN サーバソフトウェアを用いて複数の VPN プロトコルをサポートするには、システム管理者は、複数の VPN サーバソフトウェアを同時に動作させる必要がある。また、複数の VPN サーバソフトウェアのインスタンス間のメッセージ交換は、OpenVPN の場合と同様に、OS の通信機能に依存するため、OS の設定変更が必要である。

さらに、これらの VPN サーバソフトウェアは、VPN サーバに接続してきたクライアントごとに、1 個ずつ OS の仮想 L3 インターフェイスを作成する。この仮想 L3 インターフェイスとしては、OS の Point-to-Point Protocol (PPP) [91] モジュールが使用される。PPP モジュールの作成には、管理者権限が必要である。また、VPN クライアント間の通信は OS 上で作成される PPP モジュールを経由して、OS の持つルーティング機能で実現される。システム管理者は、VPN クライアント間の通信を規制するためのパケットフィルタを使用したい場合、OS に組み込まれているパケットフィルタ機能を使用する必要がある。たとえば Linux では iptables [86]、BSD では PF [36] を利用する必要がある。この設定方法は、OS によって異なる。

L2TP には暗号化の機能がないため、通常、IPsec を用いて L2TP をカプセル化した L2TP/IPsec が使用される。IPsec の処理は、OS に付属の、または外部の IPsec プログラムを実行する必要がある。たとえば、Linux の場合、カーネルモードの IPsec 暗号通信プログラムと、strongSwan [92] や racoon [71] などのユーザモードの IPsec 鍵管理プログラムの両方が必要である。正常に動作するためには、VPN サーバプログラムの開発者が意図した IPsec プログラムが OS に別にインストールされている必要がある。また、VPN サーバプログラムと IPsec プログラムとの間での協調動作のため、規定のインターフェイスを通じた連携が必要であり、プログラムの外部依存が増える。このようなプログラム間の依存関係の解決は、パッケージマネージャを用いて解決することができる。しかし、パッケージマネージャは OS の製品やディストリビューションごとに異なるため、各 OS ごとに異なるパッケージ定義を作成しなければならない。依存先のプログラムがより新しいバージョンにアップデートされたときに、予期しない問題が発生することがある。このような依存関係の問題は、Dependency Hell [64] と呼ばれる。また、依存先のプログラムが動的リンクライブラリの場合は、DLL Hell [24] などと呼ばれる。さらに、限られたリソースで動作する組み込み機器には、パッケージマネージャを搭載できない場合が多い。

本研究において実装する統合的な VPN サーバソフトウェアでは、1 つのインスタンスのみで、L2TP、SSTP、L2TPv3、EtherIP および SEVPN などの多数の VPN プロトコルに対応することを目指す。また、VPN サーバに接続中の VPN クライアント間のメッセージ交換を、外部プログラムの持つルーティングやブリッジング機能、外部の IPsec 機能を呼び出さなくても良いようにするため、VPN サーバに含める。そして、統一された 1 回の操作で、すべての VPN プロトコルに対してセキュリティ設定を適用することができるようにする。詳しくは、第 3 章で述べる。

2.1.3 OpenWRT および DD-WRT

多くの商用または家庭用ルータなどのネットワーク装置のファームウェアは、実装手法が公開されていない。OpenWRT [81] および DD-WRT [15] は、実装手法が公開されている、Linux ベースの無線 LAN ルータのファームウェアである。これらのオープンソース無線 LAN ルータのファームウェアには、ルーティング、Network Address Translator (NAT)、ファイアウォールおよび VPN サーバ機能が搭載されている。

OpenWRT および DD-WRT では、これらの機能は、Linux のカーネルおよびオープンソースソフトウェアを組み合わせて実現されている。各機能の設定や管理は、統一された Web インターフェイスから実現可能である。この統一された Web インターフェイスが、

操作対象のモジュールを呼び出して、設定や管理を行う。OpenWRT および DD-WRT では、Point to Point Tunneling Protocol (PPTP)、Layer-2 Tunneling Procotol (L2TP) および OpenVPN プロトコルに対応した VPN サーバ機能が動作する。しかしながら、これらの VPN サーバ機能は、VPN プロトコルごとに、オープンソースの異なる VPN サーバをそのまま動作させている。PPTP サーバ機能には poptop、L2TP サーバ機能には openl2tpd、OpenVPN サーバ機能には OpenVPN サーバプログラムが使用されている。

これらの異なる VPN サーバプログラムを経由した VPN トンネル間のメッセージ交換は、Linux カーネル上のルーティングプログラムで処理されている。システム管理者が OpenWRT および DD-WRT でこれらの複数の VPN サーバ機能を有効にして使用する際には、各 VPN サーバプログラム用の設定ファイルと同等の内容を、Web 管理インターフェイスまたはコマンドラインインターフェイス (Command Line Interface: CLI) で書き込む必要がある。Web 管理インターフェイスに、簡易的な設定自動生成支援プログラムはあるが、各 VPN サーバソフトウェアを使用するためにシステム管理者にとっては、システム管理者が自分自身で Linux 上に複数の VPN サーバをインストールして使用する場合とそれほど変わらない知識が必要となる。複数の VPN サーバソフトウェアにおけるユーザ認証などのセキュリティの設定も、それぞれの VPN サーバソフトウェアの設定ファイルで行う必要がある。

さらに、OpenWRT および DD-WRT における VPN サーバ機能や、複数の VPN プロトコルへの対応機能は、Linux カーネルに深く依存しており、OpenWRT および DD-WRT を動作させることができる専用のハードウェアが 1 台必要である。OpenWRT および DD-WRT を VM で動作させることも可能であるが、この場合は VM が必要となる。そのため、OpenWRT および DD-WRT 用のハードウェアや VM のコストが発生する。VM を用いる場合、物理的なネットワークとの接続のために、ホスト OS 上での管理者権限が必要となる。

本研究で実装する統合的な VPN サーバソフトウェアにおいては、OpenWRT および DD-WRT と異なり、複数の VPN プロトコルを、単一の VPN サーバソフトウェアで統一的にサポートすることを目指す。また、Linux カーネルなどの特定の OS や OS 内のモジュールに依存しないようにすることを目指す。さらに、一般的なサーバ PC やデスクトップ PC などの OS 環境上で、VM なしに、VPN サーバソフトウェアをインストールして動作させることができるようにする。そして、OS の管理者権限を必要とせず、一般ユーザ権限で動作させることができることを目指す。詳しくは、第 3 章で述べる。

2.1.4 VyOS

VyOS は、Linux ベースのルータ OS である [99]。VyOS は、統一的な CLI を用いて、ルーティング設定や VPN 設定を記述することができる。また、VPN サーバ機能として、OpenVPN プロトコルおよび L2TP プロトコルをサポートしている。OpenVPN サーバの実装には OpenVPN ソフトウェアを、L2TP サーバの実装には xl2tpd ソフトウェアを使用している。VyOS による、これら 2 つの VPN プロトコルの設定は、CLI により行うことができる。CLI の書式は統一されており、設定の記述に関して、VPN プロトコル間の統合度合いは、OpenWRT および DD-WRT よりも高い。しかしながら、OpenWRT および DD-WRT と同様に、VyOS の動作には、専用のハードウェアまたは VM が必要であり、VPN サーバ機能を既存の任意の OS 環境上で動作させることはできない。

2.2 統合的なシステムソフトウェア

2.2.1 DeleGate

DeleGate は、複数のプロトコルに対応した統合的なプロキシサーバである [89]。DeleGate は、単に通信内容を中継するだけでなく、HTTP、FTP、Gopher、NNTP、SMTP、POP などの複数のプロトコル間の変換を行うことができる。DeleGate は、異なるプロトコル間で統一的に利用可能な、共通のリソース識別のためのビューを実現するために、マウンティングと呼ばれる機能を提供している。マウンティングにより、プロトコル間の差異を吸収している。また、使用される文字コードを、プロトコル間で自動的に変換する機能も備えている。

本研究で実装する統合的な VPN サーバソフトウェアも、同様に、プロトコル間の差異を吸収するために、共通のメッセージフォーマットとして、レイヤ 2 の Ethernet フレームを使用する。レイヤ 3 の VPN プロトコルは、自動的にレイヤ 2 に変換されて処理される。また、各 VPN プロトコルにおけるユーザ認証やアドレス割り当て等の手順の違いも、プロトコルごとに実装するアダプタで吸収し、共通の処理モジュールに引き渡すことで、統合的なセキュリティ管理を実現する。

2.2.2 ユーザレベル OS のためのユーザレベルネットワーク機能

ユーザレベル OS のためのユーザレベルネットワーク機能を実現する研究がある [25]。この研究では、ユーザレベル OS (動作させるためにホスト OS のシステム管理者権限を必

要としない軽量な VM) 内の OS が物理的なネットワーク上のサーバにアクセスしようとするときに、ホスト OS における特権を必要とする低レベルネットワークアクセスを必要とせずに、ホスト OS を経由して、外部のサーバにアクセスすることができる仕組みを実現している。この研究では、通常は OS のカーネル内に実装されている TCP/IP スタックと同等のモジュールを、ユーザレベル OS の VM プログラム内に統合している。また、TCP/IP スタックを通常とは逆さに接続し、ユーザレベル OS 内の TCP/IP スタックからの外部のホストへの通信を一旦 TCP のレイヤで終端することを実現している。そして、終端した TCP の通信について、ホスト OS 上で非特権で呼び出しが可能なソケット API を用いて、改めてホスト OS の機能を用いて外部に接続を行ない、通信を実施する。通常の VM と異なり、ユーザレベル OS のネットワーク機能を有効にするために、OS のネットワーク設定を変更する必要がなく、同じ OS 上で動作している他のアプリケーションの通信に影響を与えることを避けることができる。また、ユーザモードから呼び出される汎用的なソケット API が利用可能である様々な環境に移植をすることが容易である。

ユーザレベル OS のためのユーザレベルネットワーク機能を実現する別の研究として、Slirp がある [49]。Slirp は、Personal Data Assistant (PDA) 用の OS である Palm とシェルアカウントを組み合わせて、インターネット通信を行うためのソフトウェアである。Slirp は、パケット通信に従量制の高額な費用がかかるが、シェルアカウントには費用がほとんどかからないようなパソコン通信ネットワークにおいて、シェルアカウントのみを使用して、任意の TCP 通信を実現するために使用される。Slirp の仮想 PPP アダプタが TCP を終端し、ペイロードを結合してストリームに変換するために、BSD の実装を基にした TCP/IP のスタックがユーザモードプログラムに組み込まれている。

本研究で実装する統合的な VPN サーバソフトウェアも、VPN サーバプログラム内に、本来 OS が持つべき TCP/IP スタックや DHCP スタックなどを組み込んでいる。これらの組み込みスタックは、本研究における提案手法である L2 アダプタや、VPN サーバを非特権で実行する際に物理ネットワークへのアクセスを実現するためのユーザモード NAT 機能で使用される。この仕組みにより、OS 環境上で VPN サーバ機能を使用する際には、ソケット API のみが利用可能であれば、物理ネットワークとの通信が可能となる。また、ユーザモードから呼び出される汎用的なソケット API が利用可能である様々な環境への移植性を有する。

2.2.3 Visual Studio

Visual Studio は、Microsoft 社によって開発された、C、C++、Visual Basic および C#などの複数の言語に対応した統合開発環境である [97]。Visual Studio .NET (2002 年) 以降のバージョンでは、各言語用の開発環境が統合され、1 つの統合開発環境となった。複数の言語で同一のデバッガ画面が利用可能となり、また、ある言語で記述したプログラムを、別の言語で記述したプログラムから呼び出すような開発を行うことも容易になった。統合開発環境の GUI 上におけるデバッグ操作や、コンパイラなどの設定管理は、言語間の差異を意識させない共通のものとなった。これを実現するために、Component Object Model (COM) [102] というオブジェクト指向技術が使用されている。Visual Studio に新たなプログラミング言語を追加したい開発者は、統一された COM インターフェイスを実装するコンパイラやデバッガのエンジンを実装することにより、言語を追加できる。たとえば、IronRuby tools for Visual Studio は、Visual Studio に新たに Ruby 言語 [31] を追加する、Microsoft 社以外によって開発されたプラグインである [42]。

Visual Studio が COM インターフェイスを経由して呼び出す複数のモジュールは、Dynamic Link Library (DLL) として実装されている。Visual Studio のインストーラは、多数の DLL ファイルをコピーし、COM オブジェクトとしてレジストリに登録をする。DLL は、別の DLL に依存している場合がある。Visual Studio のインストーラは複雑となり、インストール時に時間がかかるほか、インストールやアップデートに失敗する場合がある。ユーザは、インストーラのログを確認し、その原因を発見する必要がある。しばしば、アンインストール時も問題が発生する。Microsoft 社自身が、Visual Studio が通常のアンインストーラによってアンインストールできない場合のために、コンピュータを再インストールする前の最後の手段として使用できる、より強力な Visual Studio Uninstaller をオープンソースで公開している [68]。Visual Studio Uninstaller のドキュメントには、アンインストール対象以外のバージョンのインストール済み Visual Studio の環境を破壊することになると記載されている。

本研究で実装する統合的な VPN サーバソフトウェアも、Visual Studio の複数プログラミング言語サポートと同じように、異なる複数の VPN プロトコルを統一的にサポートする。また、システム管理者は、複数の VPN プロトコルに対して、共通の設定画面で設定を行うことができる。新たな VPN プロトコルを実装しようとする場合は、新たな VPN プロトコルモジュールを実装し、VPN サーバに追加することができる。しかし、本研究で実装する統合的な VPN サーバソフトウェアでは、Visual Studio と異なり、新た

な VPN プロトコルに対応するモジュールを追加する場合に、VPN サーバプログラム全体の再コンパイルが必要である。ただし、静的リンク方式を用いる場合は、2.1.2 項で述べたように、必要なプログラムを VPN サーバのプログラムファイルに内包することができるメリットがある。このメリットのため、本研究で実装する VPN サーバソフトウェアでは、複数の VPN プロトコルをサポートする方式として、静的リンク方式を用いる。

2.2.4 Eclipse

Eclipse は、Visual Studio と同様の統合開発環境である [74]。Eclipse は、Visual Studio と同様に、複数の異なる言語を統一した GUI で扱える。Eclipse は、Visual Studio と異なり、オープンソースである。Eclipse のプラグインの 1 つである Xtext を使うことにより、Domain Specific Language (DSL) を用いたプログラミングにおいて、Eclipse のコード補完やコンパイルエラーの検出、コードへの色付けなどの開発サポート機能が使えるようになる [5]。本研究では、Eclipse と同様に、複数の VPN プロトコルを統一した GUI で扱うことができる VPN サーバソフトウェアを目指す。Eclipse で特定の言語の開発環境を構築するためには、その言語のコンパイラ、デバッガ、ライブラリ等をインストールし、Eclipse でそれらを正しく呼び出す設定が必要である。本研究では、Eclipse と異なり、1 つのソフトウェアをインストールするだけで、複数の VPN プロトコルを利用できる VPN サーバを簡単に構築できるような VPN サーバソフトウェアの実現を目指す。

2.2.5 VM のイメージの配布

インストールや設定が複雑なシステムソフトウェアを簡単に利用できるようにするため、インストールや設定が完了した状態の VM のディスクイメージを配布する手法がある。たとえば、VMware Virtual Appliance Marketplace [109] では、様々なソフトウェアをインストールした状態の VM のディスクイメージがダウンロード可能である。ユーザは VM イメージをダウンロードすることにより、様々なソフトウェアがインストールされた状態から実行を開始できる。

複数の VPN サーバソフトウェアをインストールした状態の VM イメージを作成することにより、その VM イメージをダウンロードして実行したシステム管理者は、複数の VPN サーバソフトウェアを実行することができる。この方法では、複雑なインストール操作を不要とするメリットがある。しかし、複数の VPN サーバソフトウェアを設定管理する際の操作が複数回必要である問題は解決されない。また、VPN サーバソフトウェアを経由して物理的なネットワーク上のホストと通信をするために、ホスト OS の IP ルー

ティング機能またはスイッチング機能を使用する必要がある問題も残る。さらに、VM の実行やネットワーク仮想化の使用には root 権限が必要である。VM の実行をサポートしていないカーネルバージョンや OS 環境では、VM は使用できない。たとえば、VM の利用を想定していない組み込み機器システムを使用する場合は、VM が利用できない。

本研究では、VM なしで、複数の VPN プロトコルに対応し、統一的な運用管理を行うことができる VPN サーバを実現する。

2.2.6 Docker

Docker は、Linux などの OS 上で、コンテナを作成することができるオープンソースソフトウェアである [64]。コンテナとは、特定の状態の OS 環境を静的に保存し、それを複製して再利用することができる技術である。Docker では、インストールや設定などの環境構築に複雑な手順を必要とするソフトウェアを OS 環境にインストールした状態をコンテナとして保存し、配布することができる。コンテナ内の OS 環境は、ホスト OS を共有するため、VM と比較して、省メモリ、省 CPU 時間で利用できる。VM と同様に、コンテナ内のプログラムがコンテナ外に影響を与えないようにするために、コンテナ用の軽量な仮想化技術が使用されている。ネットワークの仮想化には、Virtual Ethernet (veth) が使用される [12]。veth を使用すると、ホスト OS 上の veth と、コンテナ内の veth との間で、Ethernet の MAC フレームの交換が可能になる。UberCloud [96] では、様々なソフトウェアをインストールした状態の Docker のコンテナイメージがダウンロード可能である。

複数の VPN サーバソフトウェアをインストールした状態の Docker コンテナを作成することにより、そのコンテナをダウンロードして実行したシステム管理者は、複数の VPN サーバソフトウェアを実行することができる。この方法では、複雑なインストール操作を不要とするメリットがある。しかし、この場合も、2.2.5 項と同様の問題が発生する。

本研究で実現する統合的な VPN サーバソフトウェアは、Docker が利用できない場合でも、複数の VPN プロトコルに対応し、統一的な運用管理を行うことができる VPN サーバを実現する。また、Docker を用いた手法と異なり、本研究で実装する統合的な VPN サーバソフトウェアは、プログラムファイルを 1 つダウンロードするだけで、Docker に対応していないホスト OS 上で直接実行することができる。

2.3 検閲回避システム

2.3.1 Tor

The Onion Router (Tor) は、P2P 技術を用いて実装される、インターネットへのアクセスの匿名化を目的としたオーバーレイネットワークである [18]。Tor では、匿名性を実現するために 3 つのノードを中継して通信がなされる。

Tor のノードは、公開されている Tor リレーと非公開の Tor ブリッジに分類される。検閲ファイアウォールの管理者は公開されている Tor リレーへの通信を簡単に遮断することができる。検閲が行われている国の利用者は、Web サイト、メール、その他の方法で非公開の Tor ブリッジを探し、利用できる。しかし、同様に、検閲ファイアウォールの管理者は、多数の IP アドレスに対してスキャンを実施し、Tor ブリッジを発見し、通信を遮断できる [101] [104]。obfsproxy を使用することにより、Tor クライアントと Tor ブリッジとの間の Tor のプロトコルを暗号化したり、パケットサイズやパケット送信のタイミングを変化させたりすることにより、特徴点を隠すことができる [94]。これにより、検閲ファイアウォールの管理者に対して、トラフィックを監視するだけでは Tor の通信を発見することを困難とすることができる。しかしながら、検閲ファイアウォールの管理者は、公開されている Tor ブリッジに対して、正規の Tor プロトコルを用いて接続を試みることで、Tor ブリッジを検出することができる。現在のところ、Tor は、検閲ファイアウォールの管理者によるスキャンに対する耐性を有していない。

本研究において実装する検閲回避システムは、Tor と異なり、匿名化目的ではなく、検閲ファイアウォールをバイパスすることに特化している。通信は、1 つの VPN サーバにより中継されるだけなので、Tor よりも高速である。本研究では、VPN サーバのリストを利用者に届けるために、Tor と同様に Web ページやメールを用いる。ただし、Tor とは異なり、VPN サーバのリストに無実の IP を含める。詳しくは 4.3.2 項で述べる。また、本研究では、検閲ファイアウォールの管理者によるプローブを難しくする仕組みを導入している。詳しくは 4.3.3 項で述べる。

Tor リレーや Tor ブリッジを運用することは簡単ではない。rbox-tor は、Tor リレーや Tor ブリッジが予めインストールされた CD-ROM ブートイメージを配布するプロジェクトであり、これを仮想計算機 (Virtual Machine: VM) 上で動作させるだけで Tor リレーや Tor ブリッジを動作させることができる [87]。しかしながら、rbox-tor を VM 上で動作させるためには、VM のインストールと起動が必要である。また、VM の仮想ネットワークインターフェイスを、物理的なネットワークに接続し、グローバル IP アドレスを

割り当て、インターネットから到達可能にする必要がある。VM の動作には OS 環境を実行するためのメモリや CPU 時間が必要である。また、VM を物理的なネットワークに接続するためには、管理者権限も必要である。さらに、多くの環境では NAT やファイアウォールの内側にコンピュータがあるため、NAT やファイアウォールの設定変更による Demilitarized Zone (DMZ) の設定も必要になる。これらは、気軽に中継サーバを立ち上げたいと考えるボランティアにとって、大きな負担となるため、ボランティアの数を増やしにくい。

本研究では、ボランティアの負荷を減らすために、VM なしで動作するボランティア用の中継サーバソフトウェアを配布する。消費するメモリおよび CPU 時間は、VM を利用する場合よりも少ない。また、動作させるために管理者権限も不要である。さらに、本研究では、NAT 装置の背後でも動作するなど、ボランティアのネットワーク環境の変更を不要とするための様々な工夫も行なっている。詳しくは、4.4 項で述べる。

2.3.2 BitTorrent トラッカ

本研究で実装する検閲回避システムでは、中央のサーバでアクティブな VPN サーバのリストを管理する。この方法は、BitTorrent のトラッカと似ている [14]。BitTorrent のトラッカとは、BitTorrent のノード間を接続するための情報を提供する中央サーバである。

BitTorrent のトラッカのアドレスは公開されているため、ネットワークや ISP の管理者は、ファイアウォールにより、ユーザからトラッカへの接続を、簡単に遮断することができる。BitTorrent では、Azureus Distributed Hash Table (DHT) と Mainline DHT を用いてトラッカを DHT により実装することもできる [2] [7]。

本研究では、2 つの理由により、DHT を利用できない。1 つ目の理由は、遮断をより難しくするための手法であるキースペース・ホッピングを用いるため、VPN サーバのリストを要求してきた通信相手ごとに、回答するコンテンツの内容 (VPN サーバのリスト) を変化させる必要があるためである。2 つ目の理由は、VPN サーバ間で協調して検閲ファイアウォールの管理者による調査用パケットの発信による検証を難しくするために、サーバリストを中央の VPN 中継サーバリストを管理するサーバに集約する必要があるからである。本研究のサーバリストの管理の仕組みは、これらの点で、DHT によって実現されたトラッカよりも、中央集権的である。しかし、本研究では、4.3.3 項で述べる解決手法を用いることにより、検閲用ファイアウォールに対する遮断耐性を実現する。

BitTorrent のような P2P システムに対して、少数のユーザが大量のノードを作成する Sybil Attack [19] という攻撃手法がある。BitTorrent に対する Sybil Attack の報告もあ

る [100]。本研究で実現する検閲回避システムも、Sybil Attack による影響を受ける可能性がある。詳しくは、4.5.7 項で論じる。

2.3.3 Web MIXes

多くの研究者が、検閲耐性を実現するためのシステムに関する研究を行ってきた。Web MIXes は、匿名性の実現および検閲の防止を目的とした Web アクセス中継システムを実現するための研究である [4]。この研究のシステムでは、クライアント側に Java で実装されたローカルプロキシを置き、ローカルプロキシが、中継サーバである MIX サーバ群との間で通信を行う。MIX サーバ群は、宛先のサーバとの間の通信を行う。クライアントのローカルプロキシと MIX サーバ群との間の通信においては、ランダムな長さのダミーメッセージを挿入したり、大きなメッセージを Adaptive chop-and-slice アルゴリズムによって分割するなどして、検閲者による通信内容の識別を難しくする。

本研究で実装する検閲回避システムも、VPN プロトコルによっては、単なる HTTPS 通信に見せかけたり、UDP を使用する場合のクライアント、サーバの双方のポート番号を特徴の無いランダムなものにしたりした上で、ダミーのバイト列を付加したり、無通信時でも時々ランダムなバイト列を送受信したりして、検閲者に VPN プロトコルを使用していることを気付かれないようにする。また、本研究では、Web MIXes と比較して、プロキシではなく VPN 通信機能を提供するため、クライアントアプリケーションがプロキシに対応していない場合でも利用できる。

2.3.4 中国の Great Firewall を回避するための研究

中国政府によって運用されている検閲用ファイアウォールである Great Firewall (GFW) による検閲を回避するための研究がある [13]。この研究は、GFW が TCP のコネクションを切断しようとする際に利用するメカニズムとして、偽の TCP リセットパケット (RST) が GFW によって両端のエンドポイントに送付されることを明らかにしている。そして、エンドポイント側で RST パケットを無視することにより、GFW により検閲が行われた場合でも、コネクションが切断されないようにできることを報告している。しかし、この手法による検閲回避は、ユーザの側のクライアントコンピュータと、宛先のサーバコンピュータとの両方が RST パケットを無視するように事前設定されている必要があり、不特定多数の Web サイトにアクセスしたい場合には実用的ではない。また、GFW は、TCP の偽 RST パケットを送付する以外の方法で、IP アドレス単位で通信を遮断することもできる。

本研究で実装する検閲回避システムでは、クライアントコンピュータが中継ノードまでの間を VPN 接続し、中継ノードを経由して、宛先のサーバと通信をする方式で検閲を回避する。また、GFW によって VPN 接続が遮断されないようにするために、複数の新しい手法を実装する。詳しくは、第 4 章で述べる。

2.3.5 Infranet

Infranet は、HTTP による Web アクセスの検閲と監視を回避するための研究である [27]。Infranet では、ユーザのコンピュータ上で動作するローカルプロキシである Infranet Requester と、一般的な HTTP サーバ上で動作させることができる Infranet Responder との間で、一般的な (検閲者から見ると無害な) HTTP 通信を行うが、その HTTP 通信に、検閲者から隠したいメッセージを埋め込むことができる。その埋め込まれるメッセージを、本当にアクセスしたい目的の HTTP サーバとの間の通信のためのトンネルとして利用する。これにより、Infranet Responder を経由して、目的の HTTP サーバとの間で通信が可能となる。この手法では、Infranet Responder を検閲国外のサーバにインストールしておく必要がある。不特定多数で 1 台の Infranet Responder サーバを共有する場合、そのサーバのアドレスを検閲者が入手することができるため、共有の Infranet Responder 宛の国内からの通信は、すべて検閲用ファイアウォールで遮断することができてしまう。

本研究において実装する検閲回避システムでも、Infranet と同様に中継サーバを使用するが、ユーザと中継サーバとの間のトンネル通信が遮断されないようにするために、新たな手法を提案する。詳しくは、第 4 章で述べる。

2.3.6 キースペース・ホッピング

検閲回避のために、多数のプロキシを設置し、単一のクライアントに対しては、一部のプロキシの IP アドレスのみを回答する仕組みの検閲回避システムの研究がある [28]。この研究では、キースペース・ホッピングという手法が提案されている。キースペース・ホッピングとは、各クライアントが、サーバリストの全体のうち、疑似乱数を用いて生成されたユニークな部分集合のみを取得する手法である。これは、無線通信において、無線ノードが通信妨害を避けるために周波数ホッピング方式を用いることと似ている。キースペース・ホッピングにより、検閲者が一般のユーザを装ってプロキシサーバの IP アドレスを収集しようとしても、一部の IP アドレスしか収集できない。ユーザごとに開示される IP アドレスのリストは異なるため、検閲者から見える IP アドレスが検閲用ファイア

ウォールで遮断された場合でも、ユーザは他のサーバのいずれかに接続することができる確率が高くなる。

本研究における検閲回避システムでも、中継サーバの IP アドレスリストの回答時に、キースペース・ホッピングを使用している。疑似乱数のシード値として、クライアントのネットワークアドレスの一部を使用する。この方法により、検閲当局は、中継サーバのサーバリスト全体を取得するために、広い範囲および多くの固有の IP アドレスを必要とし、サーバリスト全体を取得するコストを高めることができる。

2.3.7 SafeWeb

SafeWeb 匿名化システムという研究がある [61]。SafeWeb の目的は、Web アクセスにおける、検閲者によるコンテンツフィルタリングの回避と、Web サーバの側におけるユーザの識別の回避の 2 つである。SafeWeb では、Web ブラウザ上で動作する JavaScript でクライアントが実装されているため、ユーザのコンピュータで特別なソフトウェアを起動する必要がない。本研究において実装する検閲回避システムは、ユーザのコンピュータで VPN ソフトウェアを起動する必要があるが、Web ブラウザだけでなく、任意の TCP または UDP プロトコルを用いた通信が可能である。

2.3.8 Telex

Telex は、プロキシサーバベースの検閲回避システムであり、クライアントが Transport Layer Security (TLS) のヘッダに、楕円曲線暗号を用いた電子透かしを埋め込む手法である [105]。Telex では、クライアントは検閲されていない Web サーバに TLS でアクセスする。検閲の対象となっていない多くの Web サーバをホストしている多数の ISP に協力を依頼し、Telex Station と呼ばれる透過型プロキシを予め設置しておく。Telex Station は、電子透かしを検出できる秘密鍵を持っており、TLS のヘッダに埋め込まれた電子透かしを検出し、本来の Web サーバではなく、トンネリングのためのサーバにパケットを転送する。検閲当局は秘密鍵を持っていないので、電子透かしの有無を検出できない。この方法は、検閲当局が、Telex に協力的な ISP のアドレスブロックごとファイアウォールで遮断してしまう可能性があるという問題がある。これを避けるためには、多数の ISP に一致団結して協力してもらう必要がある。そのようにすれば、検閲当局は多数の ISP の IP アドレスを遮断しなければならなくなり、国内から海外の Web サイトが見づらくなるので、これを躊躇することになる。しかし、Telex の手法では、Telex Station を多数配置するコストと、協力的な ISP を募る必要性の問題がある。

本研究における検閲回避システムも、Telex のように、大量の IP アドレス上で通信を中継する仕組みを利用し、検閲回避を実現する。本研究では、ISP の協力を得る代わりに、多数のボランティアの協力を得る。Telex Station を ISP のインフラに設置してもらう代わりに、簡単にインストールできる統合的な VPN サーバをボランティアの PC 上で立ち上げてもらう。

2.3.9 無料や有料の VPN サービス

無料や有料の VPN サービスは、検閲用ファイアウォールを回避するために広く使われている。たとえば、vpnsurfing.com は、中継用の VPN サーバを立ち上げ、公衆に提供している [98]。無料の VPN サービスの場合は、検閲当局はその VPN サービスの Web サイトを閲覧して、中継用 VPN サーバの IP アドレスやホスト名を発見し、検閲用ファイアウォールの遮断リストに登録することが容易に可能であるという問題がある。有料の VPN サーバの場合は、料金を支払ったユーザに対してのみ中継用 VPN サーバのアドレスが明かされるため、検閲用ファイアウォールにそのサーバが遮断対象として登録される可能性は低くなる。しかし、ユーザにとっては利用のためのコストがかかる。また、検閲当局によって IP アドレスが遮断された場合は、IP アドレスを変更し、新たな IP アドレスをユーザにメール等で周知する必要がある。検閲当局がユーザに紛れている場合は、また遮断が行われる。既存の無料や有料の VPN サービスを運営する場合は、このようないたちごっこを避けることが困難である。

本研究で実装する検閲回避システムでは、VPN サーバをボランティアのコンピュータで動作させるため、ユーザにとって利用のためのコストがかからない。また、検閲当局がユーザとして IP アドレスのリストを取得した場合のために、無実の IP アドレス混入手法および協調的スパイ発見手法により、遮断への耐性を実現する。詳しくは、第 4 章で述べる。

2.3.10 GroupVPN

いくつかの VPN サービスは、中央サーバを必要としないアーキテクチャを持つ。GroupVPN は、中央サーバを必要としない、自律的な P2P 型の VPN ネットワークを構築することを目指した研究である [106]。GroupVPN では、多数の P2P ノードによって構成されるオーバーレイネットワーク上における仮想アドレスの解決を Distributed Hash Table (DHT) によって行っている。また、中央サーバに依存しないユーザ認証を実現するために、Public Key Infrastructure (PKI) を用いた証明書の検証により認証を行って

いる。通常、PKI においては Certificate Revocation List (CRL) を中央サーバなどに掲載し、これを認証者が定期的に HTTP でダウンロードする方式により証明書の無効化を周知するが、GroupVPN では、CRL をネットワーク内で DHT を用いて共有するか、または、ブロードキャストメッセージとして全ノードに配信することで、中央サーバに依存しない特定ユーザの即時無効化機能を実現している。

本研究における検閲回避システムでは、検閲用ファイアウォールに対して耐性を高めるために、キースペースホッピングを利用するため、DHT を利用しない。しかし、検閲用ファイアウォール内のユーザが、本研究で設置する、稼働中の中継サーバの一覧を有するディレクトリサーバであるリストサーバに安定的に接続できるようにするために、間接的サーバリスト転送プロトコルを導入する。間接的サーバリスト転送プロトコルにおいて配信されるサーバリストは、多数の VPN サーバのうち到達可能ないずれかのノードを経由して間接的にユーザに配信される。この際は、GroupVPN と同様に、リストに電子証明書による署名が付されるので、信頼できないノードを経由してリストを安定的にクライアントに配信することができる。詳しくは、4.3.4 項で述べる。

2.3.11 N2N

N2N は、P2P 型の VPN を実現するための研究である [16]。N2N では、ノード上に仮想 L2 インターフェイスを作成し、Ethernet フレームを UDP にカプセル化してノード間で交換することで、VPN を実現する。N2N における P2P ネットワークは、エッジノードとスーパーノードによって構成される。スーパーノードは、グローバル IP アドレスを有するノードであり、他のスーパーノードや、スーパーノードに接続してくるエッジノードとの間のメッセージのルーティングを実現する。また、スーパーノードは、現在ネットワークに参加しているエッジノードのリストを管理する。エッジノードは、NAT の内側にあり、プライベート IP アドレスを有するノードである。スーパーノードは、複数作成することができる。複数のスーパーノードを経由して、エッジノード同士のメッセージ交換が可能である。エッジノードは、いずれかのスーパーノードに接続するだけでよい。N2N を利用する場合は、ユーザは、スーパーノードを自分で立ち上げるか、他の人によって立ち上げられたスーパーノードを利用させてもらう必要がある。ユーザは、スーパーノードの IP アドレスを知っている必要があるため、N2N を用いて公開 VPN 中継サービスを実現すると、N2N のスーパーノードの IP アドレスは、検閲当局によって発見され、遮断されてしまう問題がある。

本研究で実装する検閲回避システムにおける VPN サーバは、N2N のスーパーノード

に類似している。ただし、本研究では、ユーザの通信は 1 台の VPN サーバのみを経由する。また、本研究では多数の VPN サーバを組織化し、検閲用ファイアウォールに対して耐性を高める手法を提案する。

第 3 章

統合的な VPN サーバソフトウェア SoftEther VPN の設計と実装

第 2 章では、関連研究について述べた。第 3 章では、本研究で実装した統合的な VPN サーバソフトウェア SoftEther VPN Server について述べる。

3.1 概要

第 1 章で述べたように、クライアント OS、ルータ OS やその上で動作する VPN クライアントソフトウェアごとに、対応している VPN プロトコルが異なる。単一の VPN サーバソフトウェアでは一部の VPN プロトコルしかサポートしていない場合、サポートしたいすべての VPN プロトコルのために複数の VPN サーバソフトウェアをインストールし、同時に稼働させなければならない。また、複数の VPN サーバを同時に設定し運用するために、セキュリティ設定などを、それぞれの VPN サーバに対して設定する必要があり、管理が困難となる。

これらの問題を解決するため、本研究では、SoftEther VPN Server と呼ぶ VPN サーバソフトウェアの設計と実装を行う [77]。SoftEther VPN Server は、単一のインスタンスで多数の VPN プロトコルをサポートする。サポートする VPN プロトコルは、L2TP over IPsec、Ether IP over IPsec、OpenVPN L3 および L2、L2TPv3 over IPsec、SSTP および SoftEther VPN Protocol (SEVP: Ethernet over HTTPS を実現するプロトコル) である。これらの複数の VPN プロトコルに対応することにより、広範囲に渡るリモートアクセス用途の VPN クライアント (PC やスマートフォンなど) や拠点間接続用途の VPN デバイス (アプライアンスなど) が単一の SoftEther VPN Server のインスタンスに接続できるようになる。SoftEther VPN Server を使用することにより、システム管理者

は単一のインスタンスを稼働させ管理するだけで、複数の VPN プロトコルを統一的にサポートし運用することができる。管理者は、サポートする各々の VPN プロトコルを意識することなく、1 回の操作のみで、すべての VPN プロトコルに対して統一的に、ユーザ管理、VPN クライアントへの IP アドレスの割り当て、アクセス制御およびログの保存などの設定・運用を行うことができる。さらに、SoftEther VPN Server は、インターネットサービスプロバイダ (ISP) やクラウドサービスプロバイダなどのホスティングサービスプロバイダが顧客に VPN サービスを提供するために利用可能な、単一の VPN サーバインスタンスにおける複数の顧客向けの仮想 VPN サーバ (テナント) のホスティング機能を提供する。

この章では、SoftEther VPN Server の設計、実装および評価について述べる。SoftEther VPN Server は、VPN サーバ内において、IPv4、IPv6 および Internetwork Packet Exchange (IPX) 等の任意のレイヤ 3 パケットを含む MAC フレームを交換することができるソフトウェアベースのレイヤ 2 スイッチを実装している。レイヤ 2 スイッチを用いて、レイヤ 2 の VPN プロトコルおよびレイヤ 3 の VPN プロトコルの両方を統一的に扱うために、本研究で提案する手法である、L2 アダプタと呼ばれるアダプタモジュールを用いて、レイヤ 3 パケットをレイヤ 2 フレームに変換する。

第 3 章の構成は次のとおりである。3.2 項では、複数の VPN プロトコルおよびマルチテナント機能を提供する上での問題について述べる。3.3 項では、SoftEther VPN Server の設計について、3.4 項では実装について述べる。3.5 項では SoftEther VPN Server の性能評価について述べる。3.6 項は、第 3 章のまとめである。

3.2 既存の VPN サーバソフトウェアの問題

1.1 項で述べたように、既存の VPN サーバソフトウェアの実装手法には、複数の VPN プロトコルをサポートする際における管理の問題がある。VPN サーバの管理者は、多くの種類の VPN クライアントをサポートしたり、多様な環境に適合した VPN 接続を実現したりするために、広く利用されている VPN プロトコルを複数サポートしなければならない。そのため、従来の VPN サーバソフトスウェアを用いる場合は、複数の VPN サーバソフトウェアを組み合わせて使用しなければならない。

3.2.1 多数の VPN プロトコル

広く利用されている VPN プロトコルには、表 3.1 のようなものがある。これらの VPN プロトコルは、レイヤ 3 およびレイヤ 2 に分類することができる。レイヤ 3 VPN プロト

コルは IP データグラムの交換を実現し、レイヤ 2 VPN プロトコルはローカルエリアネットワーク (LAN) で使用されている Ethernet のフレームの交換を実現する。

レイヤ 2 VPN プロトコルには、レイヤ 3 VPN プロトコルにはないいくつかの利点がある。まず、PC をレイヤ 2 VPN プロトコルを用いてネットワークに VPN 接続すれば、IP プロトコルだけではなく、非 IP プロトコル、例えば Microsoft の NetBIOS Extended User Interface (NetBEUI) [8] や Novell の Internetwork Packet Exchange (IPX) / Sequenced Packet Exchange (SPX) および Apple Talk [90] 等を使用することができる。次に、レイヤ 3 VPN プロトコルを用いる場合に必要なルーティングやプロキシ ARP [69] 等の複雑な仕組みを使用しなくても、リモートアクセスおよび拠点間接続を容易に実現できる。拠点間接続においては、両方の拠点のコンピュータを同一の IP ネットワークに所属させることができ、管理が容易になる。最後に、レイヤ 2 VPN においては、マルチキャストおよびブロードキャストベースのプロトコルをそのまま利用することができる。例えば、Multicast DNS (mDNS) [11] プロトコルを用いてマルチキャスト対応のサービスを検索したり、NetBIOS over TCP/IP プロトコルや Link-Local Multicast Name Resolution (LLMNR) [43] プロトコルを用いることで遠隔拠点のコンピュータの名前解決を行った

表 3.1: 広く利用されている VPN プロトコルの一覧

	レイヤ 3(IP パケット) を伝送する VPN プロトコル					レイヤ 2(MAC フレーム) を伝送する VPN プロトコル			
機能	IPsec トンネルモード	L2TP/IPsec	PPTP	SSTP	OpenVPN L3	OpenVPN L2	L2TPv3/IPsec	EtherIP/IPsec	SEVP
リモートアクセス		✓	✓	✓	✓				✓
暗号化	IPsec	IPsec	MPPE	SSL	SSL	SSL	IPsec	IPsec	SSL
下位レイヤ	ESP/UDP	ESP/UDP	GRE	HTTPS /TCP	TCP, UDP	TCP, UDP	ESP/UDP	ESP/UDP	HTTPS /TCP
HTTP プロキシを通過				✓	✓	✓			✓
ステートフルファイアウォールを通過				✓					✓
Windows 上のクライアント	サードパーティ	ネイティブ	ネイティブ	ネイティブ	サードパーティ	サードパーティ			サードパーティ
Mac OS X 上のクライアント	ネイティブ	ネイティブ	ネイティブ	サードパーティ	サードパーティ	サードパーティ			サードパーティ
iOS 上のクライアント	ネイティブ	ネイティブ	ネイティブ		サードパーティ	サードパーティ			
Android 上のクライアント	ネイティブ	ネイティブ	ネイティブ		サードパーティ	サードパーティ			
Linux 上のクライアント	実装あり	実装あり	実装あり	実装あり	実装あり	実装あり			実装あり
FreeBSD 上のクライアント	実装あり	実装あり	実装あり		実装あり	実装あり		実装あり	

SEVP=SoftEther VPN Protocol。MPPE=Microsoft Point-to-Point Encryption。ESP=Encapsulating Security Payload。GRE=Generic Routing Encapsulation。HTTPS=HTTP over SSL。リモートアクセスとは、VPN クライアントへの IP アドレス割り当て機能を有する VPN 接続が可能であることを意味する。ステートフルファイアウォールとは、SSL の通信のステートを検査し、非 SSL プロトコルを遮断する機能を有するファイアウォールを意味する。ネイティブとは、OS ベンダが OS に組み込んでいる VPN クライアントによりサポートされていることを意味する。サードパーティとは、サードパーティの提供する VPN クライアントのインストールにより利用可能となることを意味する。実装ありとは、1 つ以上のオープンソース実装が利用可能であることを意味する。

り、コンピュータを列挙したりすることができる。

表 3.1 の VPN プロトコルの多くは、リモートアクセスをサポートしている。リモートアクセスには、ユーザ認証および IP アドレスの割り当てが必要である。

表 3.1 の VPN プロトコルのうち、SSTP、OpenVPN (L3 および L2) および SEVP は、PC がリモートの VPN サーバに HTTP 対応のプロキシを経由して接続することを許容する。このうち、SSTP および SEVP は正規の SSL プロトコルを TCP コネクション上で使用するため、PC はステートフルファイアウォールを経由して VPN サーバに接続できる。ステートフルファイアウォールは Deep Packet Inspection (DPI) [51] 機能により、TCP ストリームが真正な SSL 通信であるかどうか見分け、SSL 通信に偽装した非 SSL 通信をブロックすることができる。

本研究では、OS 組み込み VPN クライアントプログラムで対応している VPN プロトコルを、**ネイティブ VPN プロトコル**と呼ぶ。表 3.1 に列挙されているように、各クライアントデバイスは特定のプロトコルをネイティブ VPN プロトコルとしてサポートしている。例えば、Layer 2 Tunneling Protocol (L2TP) [95] over IPsec および Point-to-Point Tunneling Protocol (PPTP) [33] 等は iOS、Mac OS X および Android におけるネイティブ VPN プロトコルである。これらの VPN プロトコルに加え、Windows は Secure Socket Tunneling Protocol (SSTP) [66] をネイティブ VPN プロトコルとしてサポートしている。CentOS Linux においては、OpenVPN [80] Layer 3 (L3) をサポートしている。これらのリモートアクセス用の VPN プロトコルに加え、複数の拠点間を拠点間接続するための VPN プロトコルとして OpenVPN Layer 2 (L2)、Layer 2 Tunneling Protocol version 3 (L2TPv3) [53] および EtherIP [34] がある。OpenVPN L2 は CentOS に、L2TPv3 は Cisco IOS に、EtherIP は NEC IX ルータにおいてネイティブ VPN プロトコルとしてサポートされている。

社内 LAN 管理者は、サポートする必要があるデバイスや実現したい機能等の要求に基づき、サポートするべき 1 個または複数の VPN プロトコルを選択する必要が必要がある。複数の VPN プロトコルをサポートする必要がある例として、たとえば、iOS のデバイスからの VPN 接続を受付けるために L2TP をサポートし、PC からの VPN 接続において mDNS をサポートする目的でレイヤ 2 通信のために OpenVPN L2 をサポートする必要がある場合等が挙げられる。

複数の VPN プロトコルのサポートは、ホスティングサービスプロバイダにおいて重要である。サービスプロバイダは、顧客の要求を満たすために、できるだけ多くの VPN プロトコルをサポートする必要がある。

3.2.2 既存の VPN サーバソフトウェアで複数の VPN プロトコルをサポートする際の問題

必要な複数の VPN プロトコルの組み合わせによっては、単一の VPN サーバによってこれら複数の VPN プロトコルをサポートすることができることがある。例えば、L2TP、PPTP および SSTP をサポートしたい場合は、Microsoft Windows に搭載されている Routing and Remote Access (RRAS) を用いてこれらの VPN プロトコルをサポートできる。しかしながら、ある 1 つの VPN サーバが、必要とする複数の VPN プロトコルすべてをサポートしていない場合は、2 個またはそれ以上の異なる VPN サーバプログラムを同時に稼働させる必要が生じる。例えば、L2TP および OpenVPN をサポートしたい場合は、RRAS と OpenVPN Server の両方を同時に稼働させる必要が生じる。複数の VPN サーバを同時に稼働させる場合は、以下のような問題が発生する。

1. VPN サーバの管理が難しくなる。ネットワーク管理者は IP アドレスの割り振り、ユーザ管理、アクセス制御およびログの保存を VPN サーバの管理業務の一部として実施しなければならない。ユーザインターフェイスや設定ファイルの書式は、VPN サーバ毎に異なる。例えば、Microsoft RRAS と OpenVPN とでは、IP アドレスの割り振りやユーザ管理の設定のためのユーザインターフェイスが異なる。VPN サーバの数が増加すると、管理に要するコストも、それに応じて増加する。
2. より多くのハードウェア資源が必要となる。単一のプラットフォームにおいて 2 個またはそれ以上の VPN プロトコルのサーバ機能を稼働することができない場合、2 個またはそれ以上の物理的なハードウェアが必要となる。例えば、L2TPv3 と SSTP の両方をサポートしたい場合は、Cisco ルータと Windows サーバ PC とを購入しなければならない。
3. 異なる VPN プロトコル間の通信速度が低下する。2 台の VPN クライアントが 2 個の異なる VPN サーバプログラムに接続されているとき、パフォーマンスが低下する。これは、2 個の VPN サーバプログラム間におけるメッセージコピーおよびコンテキストスイッチの回数の増加が原因で発生する。

広く利用されている VPN プロトコルすべてをサポートする単一の VPN サーバソフトウェアは存在していない。広く利用されている VPN プロトコルすべてをサポートする単一の VPN サーバソフトウェアを新たに制作すれば、上記の問題を解決することができる。

3.2.3　マルチテナント仮想ホスティングサービスの提供時の問題

ホスティングサービスプロバイダは、Web やメール等のサービスを、サーバソフトウェアに実装されているマルチテナント仮想ホスティング機能を用いて実現していることが多い。広く利用されている Web サーバやメールサーバは、仮想ホスティングを実装している。仮想ホスティングとは、単一のサーバに複数の内部インスタンスを持たせ、複数の顧客に対して論理的に分離した仮想専用サーバを提供することである。

例えば、Apache HTTP サーバ [72] や Postfix メールサーバ [93] はマルチテナント仮想ホスティング機能を有する。複数の顧客の複数のドメインを単一の HTTP サーバやメールサーバでホストし、顧客ごとにユーザリストやコンテンツを分離することができている。顧客の数が増加しても、その都度、割当グローバル IP アドレスを増加させたり、新たにサーバソフトウェアをインストールしたりする必要がないため、サービスプロバイダはコストを節約でき、集約効果を実現することができる。

これと同様に、もし単一の VPN サーバがマルチテナント仮想ホスティング機能を有するならば、多数の顧客ごとに分離された VPN サーバ機能を、単一の割当グローバル IP アドレスを共有させて提供することができ、また、インストールし管理する VPN サーバソフトウェアは 1 個で良いことになる。これにより VPN サービスプロバイダはコストを節約でき、集約効果を実現しつつ、顧客に代わって VPN サーバを運用することができる。しかしながら、既存の VPN サーバはマルチテナント仮想ホスティング機能を有していない。

仮想ホスティング機能を実現するためには、VPN サーバは以下の機能を提供しなければならない。

1. 分離されたユーザ管理。各テナントは、VPN 接続に係るユーザ認証の設定について、それぞれ独立した認証領域を持たなければならない。各テナントの顧客は、ユーザの新規作成や既存ユーザの削除等の操作を、サービスプロバイダに毎回依頼せずに行うことができる必要がある。
2. ネットワークの分離。ある顧客の VPN クライアントは、別の顧客の VPN クライアントと通信できてはならない。
3. 設定およびログの分離。顧客は、VPN サーバに接続したデバイス間の通信に適用されるアクセスコントロールリスト (ACL) やその他のネットワークパラメータを設定したい場合がある。同様に、顧客は、デバイスからの VPN 接続のログや通信の

ログを閲覧したい場合がある。これらの設定や保存されるログは、顧客間で分離されていなければならない。

既存の VPN サーバソフトウェアでマルチテナント仮想ホスティングに必要なすべての機能を提供するものは存在しない[*1][*2]。また、たとえ従来の VPN サーバソフトウェアの複数のインスタンスを立ち上げたとしても、OS の制約により、マルチテナント仮想ホスティングを実現することは困難である。例えば、2.1.2 項で述べたように、多くの VPN サーバソフトウェアでは、OS 上に仮想 L3 インターフェイスを作成し、OS のルーティング機能に依存して VPN クライアント間の通信を実現している。たとえば Linux では仮想 L3 インターフェイスである PPP インターフェイスの名前空間を分離することができない。ネットワークの分離ができないので、仮想ホスティング機能を実現することができない。

3.2.4 外部プログラムへの依存に関する問題

従来の VPN サーバソフトウェアは、1 つの VPN プロトコルのみをサポートする場合であっても、複数のプログラムを協調動作させなければならないものが多い。たとえば、2.1.2 項で述べた Poptop や openl2tpd は、VPN サーバ本体のプログラムが、PPP モジュールのプログラムを実行し、PPP インターフェイスを作成する。また、パケットフィルタを設定したい場合は、外部プログラムである iptables や PF を実行する必要がある。さらに、L2TP サーバプログラムは、IPsec の暗号化処理を、外部のプログラムを起動することにより行っている。

このような外部プログラムとの協調動作の必要性のため、従来の VPN サーバソフトウェアを動作する際には、必要な外部プログラムを調べ、それらを事前にインストールをしなければならない。

また、この問題は、従来の VPN サーバソフトウェアを拡張して、たとえば検閲回避システムなどの VPN アプリケーションを実装するときにも生じる。実装した VPN アプリケーションのインストーラに、すべての外部プログラムを含めることも考えられる。しかし、この場合、インストーラが複雑になり、いずれかの外部プログラムのインストール時にエラーが発生した場合には、VPN アプリケーションのインストールが失敗する。インストール時のエラーが検出できなければ、インストールは成功したように見えるが、VPN アプリケーションは正しく動作しない。VPN アプリケーションのユーザは、これらの原

[*1] 3.5.7.3 項で述べる PacketiX VPN は除く。

[*2] 一部の機能を提供するものは存在する。たとえば、Cisco 社の IOS のサービスプロバイダ向けバージョンは、Virtual Routing and Forwarding (VRF) に対応しており、複数の VPN 顧客を 1 台の Cisco ルータハードウェアに収容し、顧客ごとの通信を分離できる。

因を、依存している外部プログラムのログを精査したり、外部プログラムを個別に実行してみたりして、トラブルシューティングしなければならない。

3.3 SoftEther VPN Server の設計

3.2 項では既存の VPN サーバにおける 3 つの問題について述べた。まず、複数 VPN サーバの同時稼働は、ネットワーク管理を複雑にさせる。次に、既存の VPN サーバはマルチテナント仮想ホスティング機能を有していない。そして、外部依存プログラムが多いことにより、通常のインストールや、VPN サーバを拡張して VPN アプリケーションを作成する際に問題が生じる。この節では、これらの問題を解決するための本研究における SoftEther VPN Server の設計について述べる。

節の構成は次のとおりである。3.3.1 項では、SoftEther VPN Server がサポートする VPN プロトコルについて述べる。3.3.2 項では、VPN サーバの基本的および発展的な機能を概説する。3.3.3 項では、レイヤ 2 およびレイヤ 3 の VPN プロトコル間の相互変換について述べる。3.3.4 項では、レイヤ 2 およびレイヤ 3 の間でメッセージを変換するためのアダプタにおける ARP [84] および DHCP [21] の処理について述べる。3.3.5 項では、VPN から物理ネットワークへのアクセスについて述べる。最後に、3.3.6 項では、SoftEther VPN Server におけるマルチテナント仮想ホスティング機能について述べる。

3.3.1 サポートされている VPN プロトコル

SoftEther VPN Server の設計においては、広く利用されている以下のデバイスに組み込み VPN クライアントとして実装されている VPN プロトコルをサポートすることを目標とした。

1. PC (Windows, Mac OS X, Linux および BSD)
2. スマートフォンおよびタブレット端末 (iOS および Android)
3. 企業向けルータ (Cisco, Extreme Networks, Brocade, F5 Networks, その他)

すべてのクライアントデバイスをサポートするために、本研究では、SoftEther VPN Server に以下のようなプロトコルをサポートさせることにした。

1. レイヤ 3 VPN プロトコル
   L2TP/IPsec, SSTP および OpenVPN L3

2. レイヤ 2 VPN プロトコル
   OpenVPN L2, L2TPv3/IPsec, EtherIP/IPsec および SEVP

SoftEther VPN Server においては、PPTP のサポートを除外した。PPTP は表 3.1 にあるように広く利用されている VPN プロトコルの 1 つである。しかし、本研究におけるターゲットとするデバイスはすべて L2TP と PPTP の両方をサポートしており、PPTP のサポートを除外することによりサポートするデバイス数が減ることにはならない。加えて、PPTP には既知の脆弱性があり、代替として L2TP の使用が推奨されている [67]。サポートするデバイスが減ることにつながらないこと、およびユーザ認証や IP アドレス割り当てのプロトコルが標準化されていないことを理由に、IPsec トンネリングモードのサポートも除外した。

SoftEther VPN Server においては、レイヤ 2 VPN プロトコルとして、Ethernet を対象とする VPN プロトコルのみをサポートすることにした。他のレイヤ 2 プロトコルを対象とする VPN プロトコルは、サポートしていない。広く使われている他のレイヤ 2 プロトコルには、Asynchronous Transfer Mode (ATM) [20] や Synchronous Optical Networking / Synchronous Digital Hierarchy (SONET/SDH) [83] などがある。しかし、これらは通信キャリア網の大規模バックボーンネットワークでは使用されているものの、一般企業のネットワークではそれほど使用されていない。また、ATM や SONET/SDH 内に Ethernet のトンネルを確立し、扱いやすい Ethernet フレームを伝送することが多い。さらに、大規模バックボーンネットワークにおいてはより扱いやすい Virtual eXtensible Local Area Network (VXLAN) [58] を用いるケースもある。VXLAN は Ethernet を UDP にカプセル化してフルメッシュの VPN を構築する技術である。これらの状況を判断し、レイヤ 2 VPN プロトコルとして Ethernet を対象とする VPN プロトコルのみをサポートすれば十分であると考えた。なお、VXLAN の利用は将来増加すると考えられるため、SoftEther VPN Server には、将来的に、VXLAN のサポートも追加したい。VXLAN は、SoftEther VPN Server が対応している L2TP/IPsec などのコネクション型の VPN プロトコルではなく、コネクションレス型の VPN プロトコルである。そのため、プロトコルモジュールとして VXLAN を追加するのではなく、SoftEther VPN Server に別に実装されている遠隔の仮想 HUB 間を接続する機能を拡張して実装するのが望ましい。

3.3.2 SoftEther VPN Server の機能

SoftEther VPN Server は以下のような基礎的な VPN サーバ機能を提供する。

1. リモートアクセスおよび拠点間接続の VPN 通信機能。
2. ユーザ管理機能。ユーザの追加、削除、Remote Authentication Dial In User Service (RADIUS) [88]、Microsoft Active Directory および公開鍵認証基盤 (PKI) [62] のサポートを含む。
3. 既存の外部の DHCP サーバ、または SoftEther VPN Server に組み込まれている DHCP サーバを用いた IP アドレスの割り当て機能。クライアントデバイスがレイヤ 2 およびレイヤ 3 のいずれでも、IP アドレスを割当てることができる。
4. VPN サーバが物理的に接続されている LAN へのレイヤ 2 ブリッジまたはレイヤ 3 ルーティング機能。

SoftEther VPN Server は、さらに、以下のような発展的な VPN サーバ機能を提供する。

1. フォールトトレランスおよびロードバランス機能。複数のサーバをグループ化し、1 台のサーバで障害が発生したときは、そのサーバの接続されていた VPN セッションは他のサーバにハンドオーバされる[*3]。
2. 基本的なアクセスリスト機能。多くのルータや L3 スイッチに搭載されているパケットフィルタ機能と同等のものである。VPN サーバを流れる IPv4 および IPv6 パケットのヘッダを検査し、アクセスコントロールリスト (ACL) に従ってパケットフィルタリングを行う。
3. 拡張的なセキュリティポリシー機能。一部の高性能な L3 スイッチに搭載されているセキュリティ機能と同等のものである。基本的なアクセスリスト機能では検査することができない、パケット内のセキュリティに関わるフィールドの値を検査し、遮断するか否かを決定する。例えば、ARP や DHCP の偽装パケットを遮断したり、DHCP スヌーピングを実施して DHCP によって割り当てられた IP アドレスの使用を強制したりする。
4. トラブルシューティングや監査のためのパケットモニタリングや、必要な種類のパケットのヘッダまたはペイロードすべてをログに保存するロギング機能。

[*3] 本機能は VPN プロトコルである SEVP におけるリダイレクション機能を用いて実装されている。その他の表 3.4 の VPN プロトコルには、リダイレクション機能がない。

5. 物理的な 1 個の IP アドレスを複数の VPN クライアントで共有するためのネットワークアドレス変換 (NAT) [23] 機能。
6. コマンドラインおよびグラフィカルユーザインターフェイス。日本語、英語および簡体字中国語版が提供されている。
7. IP 優先度機能。IP ヘッダの Quality of Service (QoS) フィールドに応じて優先キューと非優先キューに振り分ける。

これらの機能は、商用製品である PacketiX VPN Server 3.0 から派生したものである。PacketiX VPN 3.0 は VPN プロトコルとして SEVP のみをサポートしていた。今回新たに設計、実装した SoftEther VPN Server は、PacketiX VPN 3.0 に複数の VPN プロトコルのサポートを追加したものである。

3.3.3 レイヤ 3-2 間の変換

SoftEther VPN Server は、3.3.1 項で述べたいくつかの VPN プロトコルをサポートする。これらのプロトコルは、レイヤ 2 およびレイヤ 3 に分類することができる。これらの異なるレイヤの VPN プロトコルを同時に取扱うためには、何らかの形でのレイヤ変換が必要となる。例えば、VPN サーバがレイヤ 3 VPN プロトコルである L2TP トンネルを経由して受信されたとき、受信されるメッセージはレイヤ 3 パケットである。これをレイヤ 2 VPN プロトコルである OpenVPN L2 トンネルに対して送信するときには、レイヤ 3 パケットをレイヤ 2 フレームに変換する必要がある。

本研究では、図 3.1 に示すように、以下の 3 つの変換手法を検討した。

手法 1

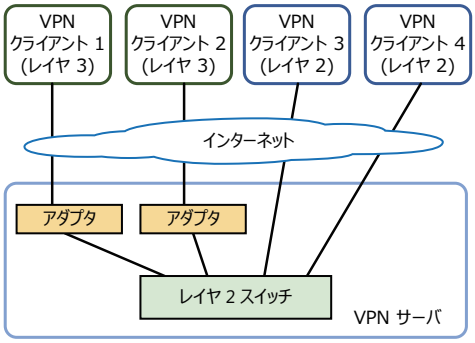
単一のレイヤ 2 スイッチを用いてメッセージを交換する。すべての受信レイヤ 3 パケットは、レイヤ 2 フレームに変換される。

手法 2

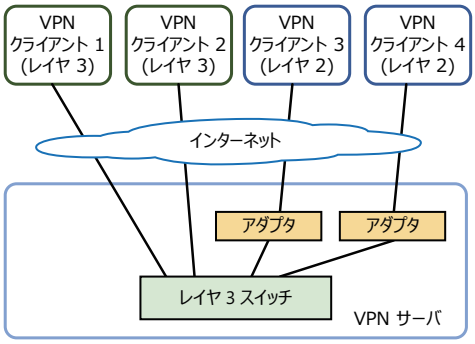
単一のレイヤ 3 スイッチ (ルータ) を用いてメッセージを交換する。すべての受信レイヤ 2 フレームは、レイヤ 3 パケットに変換される。

手法 3

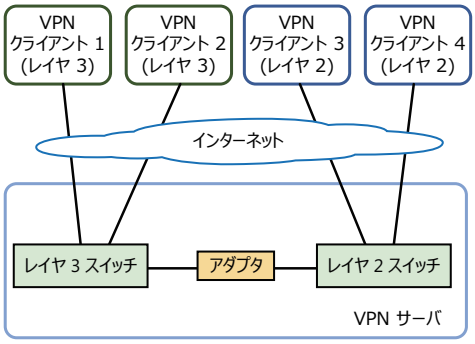
レイヤごとにスイッチを用いてメッセージを交換する。レイヤ 2 フレームのためにはレイヤ 2 スイッチを、レイヤ 3 パケットのためにはレイヤ 3 スイッチを用いる。この 2 つのスイッチの間は、レイヤ変換機構によって接続される。



(手法 1) 単一のレイヤ 2 スイッチを用いてメッセージを交換



(手法 2) 単一のレイヤ 3 スイッチを用いてメッセージを交換



(手法 3) レイヤごとにスイッチを用いてメッセージを交換

図 3.1: レイヤ 3 パケットとレイヤ 2 フレームの間の変換のための 3 つの手法

上記 3 手法のうち、手法 3 を選択すると、プログラムの実装が複雑になる。例えば、3.3.2 項で述べたアクセス制御やログ保存の機能を実装する場合を考えてみる。手法 1 または手法 2 の場合は、これらの機能はレイヤ 2 スイッチ内部にのみ実装すれば良い。一方、手法 3 の場合、これらの機能をレイヤ 2 スイッチおよびレイヤ 3 スイッチの両方に実装しなければならない。

また、手法 2 を選択すると、レイヤ 2 VPN プロトコルを正しくサポートすることができなくなる。例えば、社内ネットワークにおいて拠点間通信のために L2TPv3 や EtherIP

などのレイヤ 2 VPN プロトコルを使用したい場合がある。同一 L2 セグメントに収容されていなければ動作しないプロトコルやアプリケーションを遠隔地間で使用したい場合などである。手法 2 は、このようなプロトコルの利用を不能にしてしまう。

一方、手法 1 は、手法 3 と比較して、プログラムの実装が簡単であり、また、手法 2 と比較して、レイヤ 2 VPN プロトコルを正しくサポートすることができるようになる。そこで、本研究においては、手法 1 を選択した。

3.3.4 レイヤ変換モジュールにおける ARP および DHCP の処理

3.3.4.1 L2 アダプタ

3.3.3 項で述べたように、SoftEther VPN Server はレイヤ 3 パケットをレイヤ 2 フレームに変換する。このような変換を実施するモジュールを L2 アダプタと呼ぶ。図 3.1 の手法 1 に示すように、L2 アダプタのインスタンスは、接続中の VPN クライアントごとに作成される。L2 アダプタが VPN クライアントからレイヤ 3 パケットを受信したときは、L2 アダプタはそれをレイヤ 2 フレームに変換し、レイヤ 2 スイッチに転送する。同様に、レイヤ 2 スイッチが VPN クライアントへの送信のために L2 アダプタにレイヤ 2 フレームを転送したときには、L2 アダプタはこれをレイヤ 3 パケットに変換し、VPN クライアントに送信する。

L2 アダプタの重要な機能は、ARP の処理、IP パラメータ管理、IP ルーティング、L2 ヘッダの追加と削除、および DHCP の処理である。これらは、それぞれ、一般的な OS のカーネル内における IP 層のプロトコルスタックとして実装されている。ARP の処理は、IP パケットを L2 フレームに変換する際に必要である。ARP 要求パケットを送付する他に、一定時間、要求中の ARP 状態を管理し、ARP が未解決の宛先に対する送信キューを維持したり、タイムアウトが発生したときにそのキューを解放したりする処理が必要である。IP パラメータ管理は、仮想的な IP ホストとして動作する際の IP アドレスやサブネットマスク、デフォルトゲートウェイ、ルーティングテーブル等の情報を管理するために必要な処理である。IP ルーティングの処理は、ルーティングテーブルの参照、ネクストホップの L2 アドレスの決定、MTU の計算、フラグメントや結合などが必要である。L2 ヘッダの追加と削除は、ARP 処理や IP ルーティングと連携して行う必要がある。DHCP の処理は、UDP レイヤに位置するので、UDP スタックを必要とし、また、IP パラメータ管理のデータを参照したり書き換えたりするための連携処理も必要である。

L2 アダプタには、一般的な OS のカーネル内における IP 層のプロトコルスタックに実装されている一部の機能は不要である。たとえば、TCP の処理は行わないので、TCP ス

タックは不要である。また、パケットフィルタやセキュリティポリシーなどのセキュリティ機能も、L2 アダプタの先にある仮想 HUB に実装されているため、L2 アダプタへの実装は不要である。セキュリティ機能については、3.4.4 で述べる。

本研究の L2 アダプタで変換対象となる L3 のプロトコルは、IPv4 のみとした。その理由は、VPN を用いたリモートアクセス先の社内 LAN では IPv4 が一般的に利用されているためである。また、リモートアクセスで使用される L3 VPN プロトコルのクライアントもすべて IPv4 をサポートしているが、IPv6 はサポートしていないものもあるためである。そして、L3 VPN プロトコルにおける IPv6 の IP アドレス割り当て方法が標準化されていないためである。将来的に L2 アダプタで IPv6 をサポートする方法は、IPv4 をサポートする方法と同一である。ただし、アドレス解決のためには、ARP の代わりに IPv6 Neighbor Discovery を使用する。アドレス割り当てのためには、DHCP の代わりに、IPv6 Router Advertisement を使用する。

3.3.4.2 ARP の処理

ARP の処理においては、L2 アダプタは、通常の PC における OS の IP スタックおよびネットワークインターフェイスカード (NIC) の役割を果たす。図 3.2 に、通常の Ethernet LAN と、SoftEther VPN Server における、ARP 処理の比較を示す。図 3.2 (a) では、PC1 と PC2 の 2 台の PC がレイヤ 2 スイッチに接続されている。各 PC には、IP スタック、ARP 処理モジュールおよび NIC がある。また、各 PC は IP アドレスおよび MAC アドレスを有する。

この図において、PC1 はレイヤ 3 パケット (IPv4 データグラム) を PC2 に送信する。PC1 が PC2 にレイヤ 2 スイッチを経由してメッセージを送信する場合、PC1 は PC2 の MAC アドレスを知る必要がある。PC1 は初期状態では PC2 の MAC アドレスを知らないため、PC1 は PC2 の IP アドレスを対象とした ARP Request パケットをブロードキャスト MAC アドレスに対して送信する。レイヤ 2 スイッチは、ARP Request パケットをネットワークにあるすべての PC に配信する。

PC2 は当該 ARP Request パケットを受け取ると、ARP Reply パケットを PC1 に対して返信する。この ARP Reply パケットには、PC2 の MAC アドレスが記載されている。PC1 は、PC2 の MAC アドレスを ARP Reply パケットから抽出する。そして、PC1 は、元々送信する予定であったレイヤ 3 パケットに、宛先として PC2 の MAC アドレスを追加し、レイヤ 2 フレームを構築する。最後に、PC1 は当該レイヤ 2 フレームをレイヤ 2 スイッチに送信し、フレームは PC2 に配信される。

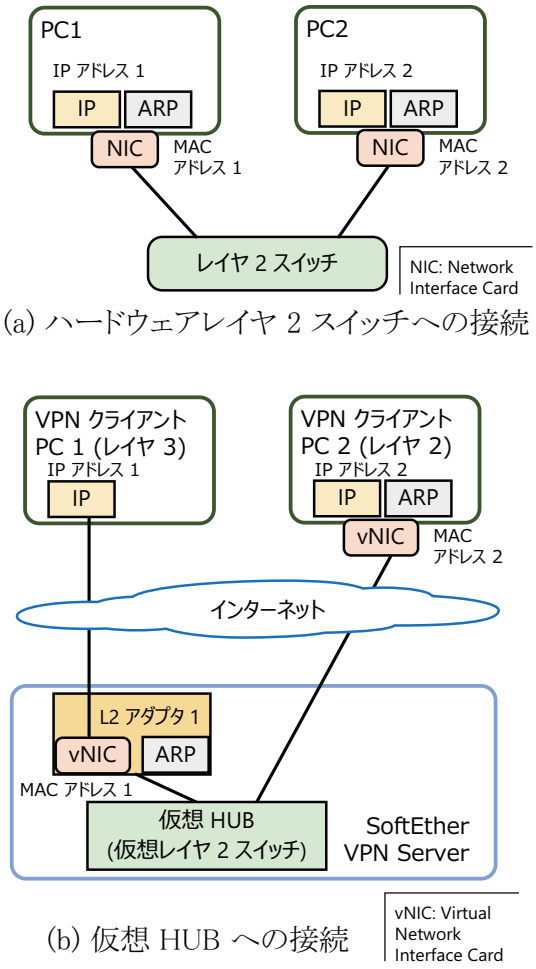


図 3.2: 2 台の PC を接続する場合におけるハードウェアスイッチング HUB と仮想 HUB との比較

一方、図 3.2 (b) は VPN サーバ内における L2 アダプタの動作を示している。SoftEther VPN Server は、MAC アドレスを宛先として用いることでメッセージを交換するための仮想的な Ethernet レイヤ 2 スイッチモジュールを有している。SoftEther VPN Server では、この仮想的な Ethernet レイヤ 2 スイッチモジュールを、**仮想 HUB** と呼ぶ。VPN サーバに接続中のリモート PC である PC1 は、レイヤ 3 VPN プロトコルによって VPN サーバに接続している。もう 1 台のリモート PC である PC2 は、レイヤ 2 VPN プロトコルによって VPN サーバに接続している。PC1 と PC2 の両方は、IP アドレスを割当てられている。PC1 と VPN サーバはレイヤ 3 パケット (IP データグラム) を交換するが、PC2 と VPN サーバはレイヤ 2 フレーム (MAC フレーム) を交換する。PC1 は TUN デバイスのようなレイヤ 3 の仮想ネットワークインターフェイスを有しており、PC2 は TAP デバイスのようなレイヤ 2 の仮想ネットワークインターフェイスを有している。レ

イヤ 2 の仮想ネットワークインターフェイスを仮想 NIC (vNIC) と呼ぶ。PC2 の vNIC は MAC アドレスを有しているが、PC1 には MAC アドレスがない。PC1 は仮想 HUB に対して L2 アダプタのインスタンスを経由して間接的に接続されており、L2 アダプタは MAC アドレスを有している。一方、PC2 は仮想 HUB に対して直接的に接続されており、L2 アダプタのインスタンスは不要である。

図 3.2 (b) において、PC1 はレイヤ 3 パケット (IPv4 データグラム) を PC2 に送信する。PC1 が VPN サーバに VPN トンネルを通じて当該パケットを送信するとき、パケットは最初に PC1 用の L2 アダプタインスタンスに到達する。パケットには L3 (IPv4) ヘッダしか含まれておらず、PC2 の MAC アドレスは記載されていない。L2 アダプタは初期状態では PC2 の MAC アドレスを知らないため、PC2 の IP アドレスを対象とした ARP Request パケットをブロードキャスト MAC アドレス宛に送信する。このときの送信元 MAC アドレスは、PC1 用の L2 アダプタインスタンスの持つ MAC アドレスである。仮想 HUB は ARP Request パケットを、接続されているすべての PC および L2 アダプタインスタンスに配信する。

PC2 が VPN トンネルを経由して ARP リエクストパケットを受信したときは、PC2 は PC1 用の L2 アダプタインスタンスの持つ MAC アドレスを宛先とする ARP Reply パケットを返信する。この ARP Reply パケットには、PC2 の MAC アドレスが記載されている。ARP Reply パケットは仮想 HUB を経由して、PC1 用の L2 アダプタインスタンスに配信される。当該 L2 アダプタインスタンスは、ARP Reply パケットを受信し、PC2 の MAC アドレスを抽出する。そして、元々送信する予定であったレイヤ 3 パケットを元にレイヤ 2 フレームを構築し、PC2 の MAC アドレス宛に送信する。このフレームは、仮想 HUB および VPN トンネルを経由して PC2 に配信される。

L2 アダプタインスタンスは、通常の PC の IP スタックにおける ARP モジュールと同様に、ARP Reply パケットを受け取ったときは、一定期間、IP アドレスと解決された MAC アドレスとの組み合わせをキャッシュに保持する。これにより、多数のレイヤ 3 パケットを送受信する際の ARP パケットの数が削減される。

SoftEther VPN Server がレイヤ 3 VPN クライアントからのパケットをレイヤ 2 VPN クライアントに対してレイヤ 2 フレームとして配信する際の仕組みは、前述のとおりである。これとは反対方向の通信が行われる場合、すなわちレイヤ 2 VPN クライアントがレイヤ 3 VPN クライアントに対してメッセージを送信しようとする場合においては、L2 アダプタインスタンスは、ARP Request パケットを受け取り、ARP Reply パケットを返信する。また、レイヤ 3 VPN クライアント同士がメッセージを交換しようとする場合は、

両方の VPN クライアント用に作られたそれぞれの L2 アダプタのインスタンス間で、同様の ARP 処理が行われる。レイヤ 2 VPN クライアント同士がメッセージを交換するときは、L2 アダプタは関係せず、両方のクライアント PC 上の IP スタックにおける ARP モジュールによって ARP 処理が行われる。

3.3.4.3 DHCP の処理

ARP 処理に加えて、DHCP 処理も L2 アダプタの重要な仕事である。VPN クライアントは、リモートアクセスにおいて接続先の LAN 内の IP アドレスを割当てられることを必要とする。IP アドレスは、多くの場合、LAN 内の既存の DHCP サーバに割り振られた IP プールから割当てられる必要がある。

レイヤ 2 VPN クライアントが VPN サーバに接続したときは、レイヤ 2 VPN クライアント上の IP スタック上で動作する DHCP クライアントが直接 DHCP Discovery および DHCP Request パケットを送信し、DHCP サーバからの DHCP Offer および DHCP Acknowledgement パケットを受信して、vNIC に IP アドレスを割当てる。これらの処理においては、L2 アダプタは必要ない。

一方、レイヤ 3 VPN クライアントが VPN 接続しようとしたときには、レイヤ 3 VPN クライアント用の L2 アダプタ上に実装されている DHCP クライアントが VPN クライアント PC に代わって DHCP Discovery および DHCP Request パケットを送信し、DHCP サーバからの DHCP Offer および DHCP Acknowledgement パケットを受信する。そして、DHCP から割当てられた IP アドレスを DHCP パケットから抽出し、これをレイヤ 3 VPN プロトコル毎に規定された IP アドレス割り当てメッセージに変換して VPN クライアントに送信する。レイヤ 3 VPN プロトコル毎に規定された IP アドレス割り当てメッセージとは、例えば L2TP over IPsec および SSTP の場合は Internet Protocol Control Protocol (IPCP) であり、OpenVPN L3 の場合は OpenVPN プロトコル中におけるメタデータメッセージである。

3.3.5 VPN から物理ネットワークへのアクセス

1.1 項で述べた問題の 1 つとして、従来の VPN サーバソフトウェアでは、VPN と物理ネットワークとをルーティング接続する際に、OS における特権が必要であるという問題がある。従来の VPN サーバソフトウェアは、VPN から物理ネットワーク上のホストに対してアクセスできるようにするために、OS が提供する (または OS に対してカーネルモ

ジュールとしてインストールする) 物理ネットワークインターフェイスブリッジ機能[*4]を利用して VPN と物理ネットワークとをブリッジ接続するか、OS 内に仮想ネットワークインターフェイスを作成し、OS の IP ルーティング機能を有効にすることにより、VPN と物理ネットワークとをルーティング接続する必要があった。これらの設定は、OS 上で動作する他のアプリケーションにも悪影響が生じる場合があり、システム管理者権限を必要とした。

SoftEther VPN Server においては、VPN から物理ネットワークへのアクセスのために、2 種類の方法を用意する。ユーザは、いずれかを選択することができる。1 つ目の方法は、ブリッジ接続である。これは、従来の VPN サーバソフトウェアにおけるブリッジまたはルーティング設定時の他のアプリケーションへの悪影響の問題を解決する。一方で、従来の手法と同様に、システム管理者権限が必要である。SoftEther VPN Server では、OS の物理ネットワークインターフェイスをオープンし、MAC フレームを送受信する処理により、ブリッジ機能を実現する。物理ネットワークインターフェイスにおける MAC フレームの送受信のコードは、OS 間で統一されていない。そこで、UNIX 系 OS では Raw socket または類似するインターフェイスを用いる。Raw socket はユーザモードから呼び出すことができる。一方、Windows では Network Driver Interface Specification (NDIS) API の呼び出しが必要となる。NDIS API は、カーネルモードからしか呼び出しが許可されていないため、API を呼び出すための小さなカーネルモードモジュールを作成し、SoftEther VPN Server の実行時に動的にカーネルにロードするようにする。これにより、カーネルモードでしか呼び出せない API をユーザモードから呼び出せるようにする。いずれも、対象となる物理ネットワークインターフェイスを、自動的に Promiscuous Mode に設定する。Raw socket の作成、およびカーネルへのモジュールのロードの両方とも、システム管理者権限が必要である。

2 つ目の方法は、ユーザモード NAT 機能である。これは、システム管理者権限が必要であるという問題を解決する。ユーザモード NAT を有効にすると、SoftEther VPN Server は、一般ユーザ権限のみで、OS 固有のネットワーク機能を呼び出したり、カーネルモジュールをインストールしたりすることなく、VPN 内のユーザが物理ネットワーク上のホストにアクセスすることができるようになる。これを実現するために、VPN 内で動作する仮想の TCP/IP スタックを用いる。これは、2.2.2 項で述べたユーザレベル OS のためのユーザレベルネットワーク機能を、VPN から物理ネットワークへのアクセスに特化

[*4] 複数のレイヤ 2 のインターフェイス間で MAC フレームを交換する機能である。スイッチと同義であるが、多くの OS ではブリッジと呼ばれている。

して実現したものである。VPN 内のユーザが、物理ネットワーク宛に TCP または UDP のパケット送信しようと、VPN サーバ内で実装されている TCP/IP スタックがこれを受信し、TCP または UDP のセッションの終端をする。TCP のパケットについては、セグメントの再結合を行ない、ストリームに変換する。UDP のパケットについては、データグラムのまま扱う。ユーザモード NAT は、TCP および UDP のいずれについても、セッションを管理し、新しいセッションが作成される際に、ソケット API を呼び出して TCP または UDP のソケットを作成する。そして、そのソケットを用いて、宛先のホストとの間で通信を行う。ソケットの利用は、Windows、Linux、Solaris、FreeBSD および Mac OS X などの広く使われているほぼすべての OS で共通して、一般ユーザ権限で可能である。そのため、ユーザモード NAT を利用する際には、SoftEther VPN Server は一般ユーザ権限で動作していてもよい。また、ソケット API が利用可能な他の OS に移植をすることが容易である。そして、一般ユーザ権限で呼び出されるソケット API のみが使用されるため、同一 OS 上で動作している他のアプリケーションの通信に大きな影響を生じさせることがなくなる。

本研究で実装したユーザモード NAT 機能には、2 つの制限事項がある。1 つ目は、家庭用ブロードバンドルータなどで実装されている NAT 機能と同様の制限事項である。VPN 内のクライアントは、外部の物理的な LAN 上のサーバの任意の TCP または UDP ポートに対し、通信を始動できる。一方、外部の物理的な LAN 上のクライアントは、VPN 内のサーバに対して、通信を始動できない。ブリッジと異なり、このような非対称となる問題があるため、用途が制限される。しかし、リモートアクセス VPN における一般的な用途である、VPN で LAN 上のサーバにアクセスをする場合は、この問題は発生しない。

2 つ目の制限事項は、UNIX 系で ICMP パケットの送受信ができないことである。一般ユーザ権限における ICMP パケットの扱いは、OS によって挙動が異なる。Windows では、一般ユーザ権限でも ICMP パケットの送受信が可能である。ところが、UNIX 系では、Raw socket を作成するために管理者権限が必要である。そのため、UNIX 系でユーザモード NAT 機能を使用する場合は、ICMP パケットの送受信はできない。この場合、VPN のユーザは、外部の物理ホストに、ping を送信したり、traceroute を実行したりできない。この問題は、ユーザから ICMP 要求パケットが送信されたときに、管理者権限で実行されるよう設定されている ping コマンドや traceroute コマンドを子プロセスとして呼び出し、その結果をパースして、ユーザへの ICMP 応答パケットとして返却する方法で解決できる。SoftEther VPN Server が対象としているすべての UNIX 系 OS では、ping および traceroute コマンドが利用可能である。しかし、これらのコマンドのオプション

や結果の書式が OS 間で統一されていないという問題がある。また、ICMP パケットごとにプロセスを立ち上げるオーバーヘッドが懸念される。現在、この方法による ICMP のサポートは未実装である。ただし、TCP および UDP と比較して、ICMP のサポートがないことは致命的ではない。

本研究で実装したユーザモード NAT 機能は、IPv4 のみに対応させた。その理由は、3.3.4.1 項で述べた、L2 アダプタを IPv4 のみに対応させた理由と同じである。IPv6 においても、将来的に、ユーザモード NAT 機能による LAN 上の IPv6 ホストへの VPN からのリモートアクセスは可能である。ただし、IPv4 の場合と同様に、NAT による非対称性が生じる。

3.3.6 マルチテナント仮想ホスティング

3.2.3 項において、VPN サーバにおけるマルチテナント仮想ホスティング機能の必要性を述べた。この節では、マルチテナント仮想ホスティングを実現するために SoftEther VPN Server に実装されている 2 つの機能について述べる。

1 つ目は、単一の VPN サーバインスタンスに複数の仮想 HUB を作成することができる機能である。単一の VPN サーバ内の複数の仮想 HUB 同士は、デフォルト状態で、互いに分離されている。ホスティングサービスプロバイダは、VPN サーバにおける最上位の権限 (VPN サーバ全体の管理者としての権限) を持つ。VPN サーバ全体の管理者のみが、仮想 HUB の作成や削除、および各仮想 HUB の管理権限の顧客への割り当てを行うことができる。顧客は、自己に割当てられた仮想 HUB の管理者権限を有する。仮想 HUB の管理者は、仮想 HUB 内にユーザを作成したり、削除したり、その他の設定を変更したりすることができる。これらの管理作業を行う際に、顧客はその都度、VPN サーバ全体の管理者に依頼をする必要がない。

2 つ目は、タグ付き VLAN によるトランキング機能である。サービスプロバイダが IaaS 型のクラウドサービスを提供しており、顧客ごとに仮想ネットワークを定義している場合、当該仮想ネットワークを集約したタグ付き VLAN [40] インターフェイスを SoftEther VPN Server の物理ホストに接続することができる。IaaS 型のクラウドサービス側の顧客ごとの仮想ネットワークに割当てられている VLAN ID と、SoftEther VPN Server 上の顧客とをマッピングすることにより、各顧客はそれぞれの IaaS サービス上の仮想ネットワークに接続することができる。この利用形態は、リモートアクセスおよび拠点間接続 VPN の両方の目的で活用できる。

3.3.7　1 つのプログラムへの各モジュールの内包

3.2.4 項において、VPN サーバが複数の外部プログラムを起動しなければならない場合の依存関係により発生する問題について述べた。SoftEther VPN Server では、この問題を、1 つのプログラムへの各モジュールの内包により解決する。

SoftEther VPN Server では、従来の VPN サーバプログラムと異なり、VPN サーバのプログラム内で、VPN サーバとしての処理を行うためのコードをすべて包含する。外部の PPP プログラムを起動して PPP の処理をさせている従来の VPN サーバプログラムと異なり、SoftEther VPN Server では、プログラム内に、PPP の処理を行うモジュールを含める。外部の IPsec プログラムを起動して IPsec の処理をさせている従来の VPN サーバプログラムと異なり、SoftEther VPN Server では、プログラム内に、IPsec の処理を行うモジュールを含める。3.3.4 項や 3.3.5 項で述べた L2 アダプタや NAT モジュールなどのプログラムも、SoftEther VPN Server のプログラム内に組み込む。IPsec や SSL [6] で使用される暗号ライブラリは、OpenSSL などの、プログラム内に組み込むことができるライブラリを使用する。

統合的な VPN サーバのプログラムにモジュールを内包するための手順を以下に示す。

1. 従来の VPN サーバソフトウェアが依存していた外部プログラムのソースコードを調査し、理解する。たとえば、IPsec プログラムについては、ユーザモード内の IPsec デーモンプログラムと、カーネルモード内の IPsec フィルタプログラムのソースコードを読む。同時に、対象となるプロトコルの規格書 (RFC など) を読む。
2. 従来の VPN サーバソフトウェアが依存している外部プログラムを呼び出す部分のコードと、呼び出される部分のソースコードを読み、関係性を理解する。
3. 依存している外部プログラムが、さらに別の外部プログラムやライブラリに依存している場合は、それらの依存先のソースコードを読む。
4. 従来の VPN サーバソフトウェアの実装者が、なぜ、外部プログラムに依存するように実装をしたのか、その理由を考える。多くの場合、その要因は、依存先に外部プログラム側にある。依存先の外部プログラムが、独立したプログラムとして動作する一方で、新たなプログラム内にそのまま組み込むことが考慮されていない作りになっているためである。
5. これらの結果を踏まえて、本研究では、従来の外部プログラムに相当する部分を、VPN サーバのプログラム内で実装する。たとえば、単に外部プログラムを実行し、

データを渡して、実行中の外部プログラムの状態を書き換えたり、結果を受け取るだけであれば、この外部プログラムに相当する部分は、スレッドとして実装する。データの受渡しは、スレッド間の共有メモリを確保してポインタを受渡しするか、プロセス内で利用できる軽量のパイプを用いて実装する。

6. 対象となるプロトコルの規格書 (RFC など) を読みながら、これまでに検討した内容に基づき、プログラムを書く。

この手法により、SoftEther VPN Server は、様々な OS が汎用的に提供するシステムコールやライブラリ以外の外部プログラムに依存しなくなる。そのため、これらの機能を有する OS に SoftEther VPN Server を簡単に移植することができるようになる。SoftEther VPN Server が依存しているシステムコールやライブラリの一覧は、以下のとおりである。

メモリ管理
: スレッドセーフなヒープメモリ割り当て関数である malloc(), free(), realloc() 関数が必要である。これは、多くの OS で利用可能である。

スレッド作成
: スレッドを作成することができるシステムコールが必要である。これは、多くの OS で利用可能である。

スレッド同期処理
: スレッド間における同期のための Condition Variables (またはイベント)、パイプおよび mutex (またはクリティカルセクション) 機能が必要である。これは、多くの OS で利用可能である。

ファイル入出力
: ログファイルへの出力や、設定ファイルの読み書きのために、open(), close(), read(), write(), lseek() のシステムコールが必要である。これは、多くの OS で利用可能である。ファイルシステムを持たない OS への移植の場合は、この部分を、他の外部記憶の読み書き (フラッシュメモリなど) に置き換える必要がある。

ソケット API
: 通信のために、ソケットの作成、削除、TCP および UDP の接続、Bind 処理、Listen 処理、Accept 処理、Poll 処理、Peek 処理、ソケットへの読み書き処理、標準的なソケットオプションの設定のためのシステムコールが必要である。これは、多くの OS で利用可能である。

zlib

zlib [110] は圧縮ライブラリである。一部の VPN プロトコルで圧縮をサポートするために使用されている。zlib はメモリ上で完結されたデータ処理を行うため、C 言語が使用できる数多くの OS に移植されている。新たな OS への移植も容易である。

OpenSSL

OpenSSL [111] は暗号ライブラリである。SoftEther VPN Server では、暗号やハッシュ関数を利用している。また、一部の VPN プロトコルで SSL トンネルを確立するために使用している。OpenSSL の暗号やハッシュ関数は、メモリ上で完結されたデータ処理を行うため、C 言語が使用できる多くの OS に移植されている。また、SSL 通信の処理では、ソケット API の呼び出しのみが必要とされており、多くの OS に移植されている。ソケット API を有する新たな OS への移植も容易である。

レイヤ 2 ネットワークインターフェイスの読み書き (オプション)

3.4.3 項で述べたように、VPN から物理ネットワークへのアクセスのためにブリッジ機能を使用したい場合は、レイヤ 2 ネットワークインターフェイスの読み書きのシステムコールを呼び出す必要がある。Windows では NDIS API をカーネルモードから呼び出す必要がある。UNIX 系 OS では Raw socket を使用する。いずれも、管理者権限が必要である。権限がない場合、またはこれらのシステムコールが用意されていない OS に移植をする場合には、3.4.3 項で述べたユーザモード NAT が、ブリッジの代替として利用できる。ただし、その場合は用途に制限が生じる。

ICMP パケットの送受信 (オプション)

ICMP パケットの送受信のためのシステムコールがある場合、3.4.3 項で述べたユーザモード NAT を使用する場合でも、ICMP パケットの送受信が可能となる。Windows は一般ユーザ権限で ICMP パケットを送受信する機能があるが、UNIX 系 OS にはない。したがって、UNIX 系 OS 上で一般ユーザ権限で VPN サーバを実行する場合、物理的な LAN 上のホストとの間での ICMP パケットの送受信はできない。

3.4 SoftEther VPN Server の実装

この節では、SoftEther VPN Server の実装について述べる。3.4.1 項では内部モジュールの概要について述べ、3.4.2 項ではこれらのモジュールがどのように複数の VPN プロトコルをサポートしているかを述べる。3.4.3 項では、本研究で用いた最適化の手法について述べる。3.4.4 項では管理機能について述べる。3.4.5 項では OS 移植性を実現するためのプラットフォーム抽象化レイヤについて述べる。

3.4.1　内部モジュール

図 3.3 は SoftEther VPN Server の内部モジュールを示したものである。最も重要なモジュールとして、以下のものがある。

1. VPN プロトコルモジュール。7 個の VPN プロトコルをサポートする。
2. VPN プロトコル間の共有モジュール。これらのモジュールは、SSL、HTTP、Point-to-Point Protocol (PPP)、IPsec およびユーザ認証を実装する。
3. 3.3.3 項および 3.3.4 項で述べた L2 アダプタおよび仮想 HUB。

図 3.4 に 7 個の VPN プロトコルモジュールが受け取った 7 種類の VPN プロトコルのメッセージが共有モジュールを経由して仮想 HUB に到達する様子を示す。例えば、L2TP/IPsec プロトコルの通信を受信した場合は、まず IPsec モジュールで IPsec の鍵交換を行ったり、IPsec のカプセル化を解除したりする。次に、L2TP モジュールで L2TP のトンネル確立処理を行ったり、カプセル化を解除したりする。そして、PPP モジュールで PPP のネゴシエーションを行ったり、PPP のカプセル化を解除したりする。この処理を経て、ユーザが送信した IP パケットを取り出す。最後に、この IP パケットを、L2 アダプタを経由して MAC フレームに変換し、仮想 HUB に渡す。

本研究では、SoftEther VPN Server を C 言語で記述した。合計プログラムサイズは、約 390,000 行である。このうち、3.3.1 項から 3.3.4 項までに述べた複数 VPN プロトコル対応機能を実装するプログラムは、約 25,000 行である。VPN プロトコル別の実装の行数は、表 3.2 の通りである。

3.4.2　VPN プロトコル固有のモジュールおよび共有モジュール

1 つの VPN サーバソフトウェアで複数の VPN プロトコルをサポートするためには、VPN プロトコルごとにプログラムを記述する必要がある。VPN プロトコル間では、共通の処理が多くある。たとえば、PPP の処理は、SSTP および L2TP で共通である。共通の処理をモジュール化し、VPN プロトコルによるデータがモジュール間を流れるようにする。このようにすれば、将来あるモジュールでバグが発見されたときに、1 箇所を修正するだけで、そのモジュールを流れるすべての VPN プロトコルに対する処理が修正され、バグ修正の際のコード変更量が削減できる。また、サポートしたい VPN プロトコルが増える場合にも、共通部分が利用できれば、新たに実装すべきコードの量が削減できる。

表 3.2: SoftEther VPN Server のうち複数 VPN プロトコル対応のためのプログラムの行数の分布

モジュール	ソースコード名	行数 (概算)
L2 アダプタ	IPsec_IPC.c	2,100 行
IPsec モジュール	IPsec.c, IPsec.h, IPsec_IKE.c, IPsec_IKE.h, IPsec_IkePacket.c, IPsec_IkePacket.h	10,700 行
L2TP モジュール	IPsec_L2TP.c, IPsec_L2TP.h	2,700 行
PPP モジュール	IPsec_PPP.c, IPsec_PPP.h	2,800 行
EtherIP モジュール	IPsec_EtherIP.c, IPsec_EtherIP.h	500 行
SSTP モジュール	Interop_SSTP.c, Interop_SSTP.h	1,300 行
OpenVPN モジュール	Interop_OpenVPN.c, Interop_OpenVPN.h	3,100 行
Windows Vista / 7 / 8 用カーネルモードモジュール	IPsec_Win7.c, IPsec_Win7.h, IPsec_Win7Inner.h, Wfp.c, Wfp.h	1,800 行
合計		25,000 行

3.4.2.1 L2TP モジュール

本モジュールは VPN プロトコルである L2TP over IPsec および L2TPv3 over IPsec を実装する。これらのプロトコルは、仕様上は 1 本の L2TP トンネル内に複数本のチャネルを確立することができる。通常の VPN クライアントは 1 本のチャネルしか使用しないが、本モジュールは将来の拡張のために複数のチャネル確立に対応している。

L2TP および L2TPv3 の仕様はシンプルなものであるが、実装の段階では複雑なプログラミングが必要とされる。具体的には、L2TP トンネルおよび L2TP チャネルの確立に必要なメッセージ交換において、TCP と同様のメッセージ再送機能の実装を要する。届いたメッセージに対する確認応答メッセージの返信処理、送信したメッセージに対する確認応答メッセージの受信を確認するまで再送キューに入れておきタイマーを用いて一定時間ごとに再送する処理、各送受信メッセージのシーケンス番号を管理し、受信メッセージについてはシーケンス番号順に並び替える処理などが必要である。

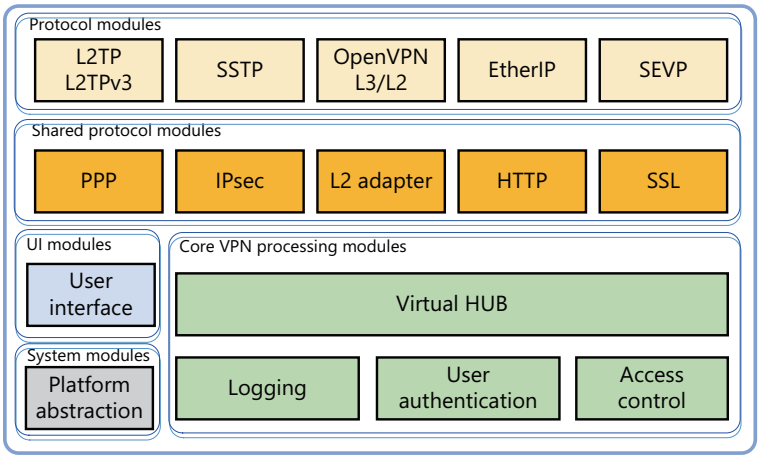


図 3.3: SoftEther VPN Server の内部モジュール

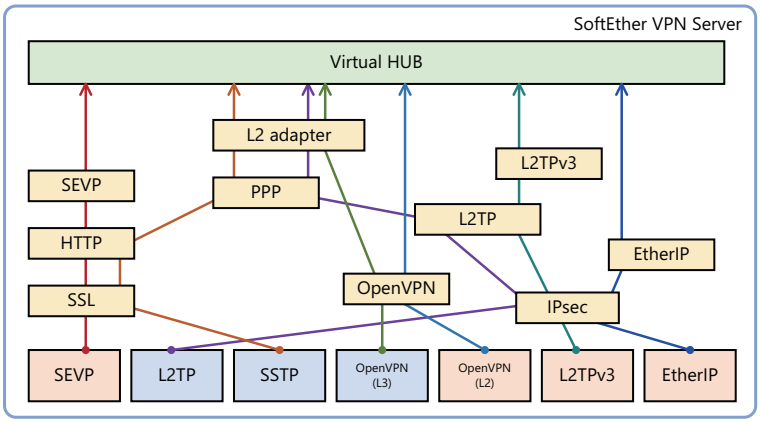


図 3.4: 7 種類の VPN プロトコルのメッセージが共有モジュールを経由して仮想 HUB に到達する様子

3.4.2.2 OpenVPN モジュール

本モジュールは OpenVPN L2 および L3 を実装する。これらのプロトコルは OpenVPN プロジェクトおよび OpenVPN Technologies, Inc. によって策定されている。これらのプロトコルは伝送レイヤとして UDP および TCP を使用する。したがって、本研究で作成したモジュールは、2 つの VPN プロトコルを 2 つの伝送レイヤ上でサポートするように実装されている。

OpenVPN プロトコル上における SSL の取扱いを実装する際には、特に努力を要した。OpenVPN プロトコルは、当初は UDP のみをサポートしており、UDP 上で SSL を用いてセキュアチャネルを確立するために、UDP 上に独自の TCP と同様のメッセージ再送機能を有していた。OpenVPN はその後に新たに TCP をサポートすることになった。した

がって、OpenVPN プロトコルは、再送機能がもともと実装されている TCP の上に UDP がカプセル化され、その UDP 上でさらに再送機能が実装されるという複雑な仕様となっている。しかし、これらの仕様は十分にドキュメント化されておらず、OpenVPN のソースコードを解析しつつ実装を行った。

OpenVPN L3 モジュールは、OpenVPN L3 クライアントに対して、接続時に IP アドレスを割当て、メタデータとしてクライアントに通知する必要がある (同時に、LAN 内で使用すべき DNS サーバの IP アドレスも通知することができる)。IP アドレスは、仮想 HUB に組み込まれている DHCP サーバ機能か、または既存の LAN 上の外部 DHCP サーバのいずれかの IP プールから取得される必要がある。IP アドレスの取得が完了しなければ、OpenVPN L3 モジュールはクライアントからの接続要求を完了することができない。したがって、OpenVPN L3 モジュールでユーザ認証が完了したら、直ちに L2 アダプタのインスタンスが作成され、DHCP Discovery および DHCP Request パケットが仮想 HUB を経由してブロードキャストされる。ところが、OpenVPN プロトコルは、既存の OpenVPN クライアント (Windows) 側の実装で独自の PPP アダプタを利用しており、PPP アダプタの利用の実装上の固有の問題により、割り当てられる IP アドレスの第 4 オクテットが、4 の倍数 +1 でなければならないという制約がある。OpenVPN L3 プロトコルは本来、OpenVPN 専用に大きな IPv4 サブネットを割当てておく利用方法を想定して開発されていたため、SoftEther VPN Server の実装上の目標である、VPN プロトコルにかかわらず共通の IP アドレス割り当ての仕組み (既設 DHCP サーバの利用) と親和性に欠けることが分かった。なぜならば、「DHCP サーバに対して第 4 オクテットが 4 の倍数 +1 となる IP アドレスをください。」というような要求はできないためである。そこで、解決策としては DHCP サーバから、第 4 オクテットが 4 の倍数 +1 となるような IP アドレスの割り当てがオファーされるまでの間は繰り返し IP アドレスを要求し、そのような IP アドレスがようやく提示された後にその他に仮に取得した全 IP アドレスを解放するという仕組みを実装した。

3.4.2.3 IPsec モジュール

本モジュールは IPsec を実装する。IPsec は、L2TP、L2TPv3 および EtherIP によって共有されている。本モジュールは IPsec v2 および Internet Key Exchange (IKE) v1 をサポートしている。IPsec v3 および IKE v2 [45] をサポートしている VPN クライアントは少数であり、これらの VPN クライアントは IPsec v2 および IKE v1 もサポートしているため、本研究では、これらのプロトコルをサポートから除外した。本モジュールは、以

下の追加的仕様も実装している。

- Negotiation of NAT-Traversal in the IKE (INTERNET-DRAFT) [47]
- Negotiation of NAT Traversal in the IKE (RFC 3947) [48]
- Traffic-Based Method of Detecting Dead IKE (RFC 3706) [38]
- Encapsulating Security Payload (RFC 4303, RFC 4305) [22] [46]
- UDP Encapsulation of IPsec ESP Packets (RFC 3948) [39]

3.4.2.4 PPP モジュール

2.1.2 項で述べたような、従来の VPN サーバと異なり、SoftEther VPN Server は、ソフトウェア内で PPP モジュールを実装している。PPP モジュールは、L2TP および SSTP のプロトコルの処理で共有されている。本モジュールにおいては、PPP における複雑なユーザ認証処理を実装した。本モジュールは、広く使用されている 2 つの認証プロトコルである Password Authentication Protocol (PAP) および Microsoft Challenge Handshake Authentication Protocol version 2 (MS-CHAPv2) をサポートしている。ユーザ認証は、SoftEther VPN Server 内部のユーザデータベースによるパスワード認証だけではなく、外部の RADIUS および Microsoft Active Directory サーバによる認証にも対応している。

VPN クライアントが PAP を用いる場合、VPN クライアントから VPN サーバへは、平文でパスワードが送信される。実際には IPsec によって暗号化されるが、これは危険であり、多くの VPN クライアントではデフォルトで無効化されている。デフォルトで有効な MS-CHAPv2 は Microsoft 独自の CHAP 認証プロトコルであるが、事実上の標準となっており、Microsoft の OS のほか、Mac OS X、iOS および Android でもサポートされている。したがって、MS-CHAPv2 への対応は必須である。ところが、SoftEther VPN Server などの外部のプログラムが Microsoft Active Directory サーバに対して MS-CHAPv2 の認証要求を転送する方法がドキュメント化されておらず、対応に苦労した。Windows 2000 Server 組み込みの RRAS サーバのプログラムを吟味することにより、LsaLogonUser API において MS-CHAPv2 の認証要求を出すための隠し構造体利用方法を発見し、これを利用することで解決した。ただし、この解決策は、Windows 上でしか利用できない。この問題を解決するため、Active Directory サーバとの間の通信において Remote Authentication Dial-In User Service (RADIUS) も利用可能な MS-CHAPv2 対応の認証クライアントモジュールを作成し、VPN サーバに組み込んだ。このモジュールは、UDP ソケットのみを使用して認証要求を出すため、SoftEther VPN Server が動作するすべての OS で動作

する。

PPP モジュールは、IP アドレスをリモートの VPN クライアントに対して割当てる役割も担っている。OpenVPN L3 プロトコルと同様、PPP の接続確立に先立って、L2 アダプタのインスタンスを作成し、DHCP サーバから IP アドレスを取得する。取得した IP アドレスは、PPP トンネル上で動作する IPCP を用いてクライアントに通知される。LAN 内で利用すべき DNS サーバの IP アドレスもオプションとして通知することができる。

3.4.2.5 Windows カーネルドライバモジュール

IPsec サーバ機能を Windows Vista およびそれ以降のバージョンの Windows 上で実装するために、カーネルハックを行う必要があった。Windows カーネルは、IPsec データグラムをネットワークインターフェイスから受け取った後、たとえそのデータグラムのために bind されているユーザプロセスのソケットが存在したとしても、そのユーザプロセスにデータグラムを渡さずに、カーネル内で自ら IPsec 処理をしてしまう。このドキュメント化されていない挙動は、ユーザプロセスが IPsec データグラムを独自に処理することを不可能にしている。

この Windows における問題を解決するため、本研究では Windows Filtering Platform 用のカーネルドライバを実装した。本ドライバは Windows カーネルが IPsec データグラムを読み取るよりも前に、IPsec データグラムのヘッダを一時的に書き換える。Windows カーネルはこれを IPsec データグラムではないと解釈し、ユーザプロセスに渡す。ユーザプロセスでは書き換えられたヘッダを元に戻して処理をする。

3.4.3 最適化手法

VPN サーバは内部に多数のメッセージキューおよびスレッドを有している。十分な考慮を行わずに実装を行うと、メッセージコピーやスレッド同期によるオーバーヘッドが大きく発生する。

SoftEther VPN Server 内部においては、OS カーネル内で用いられるものと同様の、不必要なメッセージコピーの回数の削減を行う手法を用いた。多くの OS カーネルでは、受信したメッセージや送信しようとするメッセージを複数のモジュール間で交換する。また、この際にヘッダを取り外したり取付けたりする処理を行う。この際のメッセージコピーの回数を削減するため、例えば BSD では mbuf [55]、Linux では skbuf [3]、Windows では NET_BUFFER_LIST および NET_BUFFER [65] などのデータ構造体が利用されている。SoftEther VPN Server においては、これらの既存のデータ構造体の機能を簡素化

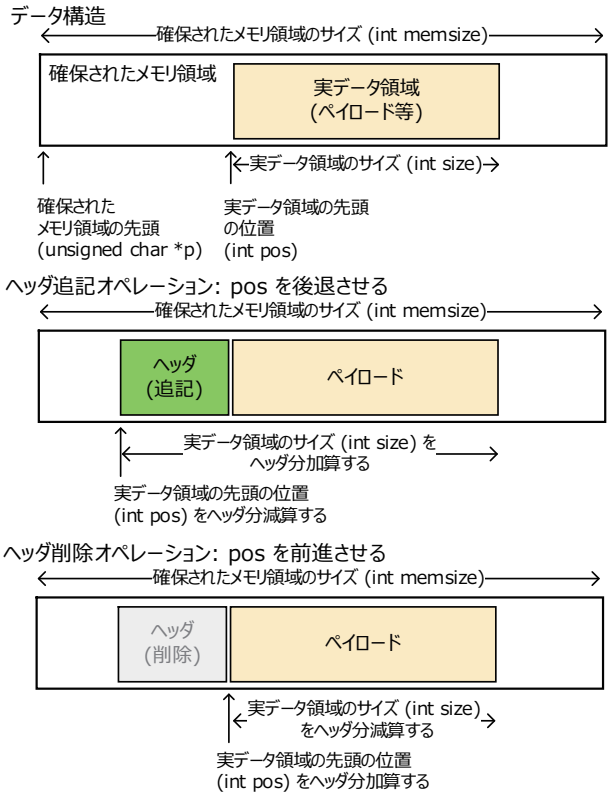


図 3.5: ヘッダの取り付け、取り外しにおいてメッセージコピーの回数を削減するためのデータ構造とオペレーション

したデータ構造体を用いた (図 3.5)。このデータ構造体では、当初必要なデータの領域の前後のメモリ領域も予約しておき、データの先頭位置および末尾位置を前進または後退させる方法により、メモリのコピーや再確保なしで、データの追記と削除ができる。本データ構造体のオペレーションでは、malloc()、free() および realloc() のみの汎用メモリ管理関数しか使用していないため、OS 間の移植性を有する。

SoftEther VPN Server は、モジュール間の同期とメッセージ交換におけるオーバーヘッドを削減するために、2 種類のメッセージキューを実装している。1 つ目はロック付きの通常のキューであり、2 つ目は高速に動作するロックなしのキューである。本研究ではまた、同一の VPN コネクションから同時に届いたメッセージを処理する際のロックの回数を削減した。これらのキューは OS の同期機能を内部で利用しているため、移植性を確保するために、プラットフォーム抽象化レイヤ内に実装されている。

3.4.4　管理機能

3.2.2 項で述べたとおり、複数の VPN プロトコルのサポートのために複数の VPN サーバを稼働させることは、設定や運用作業を複雑にする。SoftEther VPN Server では、この問題を解決するために、すべての VPN プロトコル間で、ユーザ認証やアクセスコントロール、ログ管理等を共通化している。

ユーザ認証やアクセスコントロール、ログ管理等の管理機能は、仮想 HUB に実装されている。この手法により、7 種類の VPN プロトコルに対して、統合的なセキュリティ機能を提供することができる。7 種類の VPN プロトコルは、いずれも、ユーザ認証の際に、仮想 HUB のユーザ管理機能を用いて認証を行う。7 種類の VPN プロトコルにより送受信されるメッセージは、MAC フレームとして、仮想 HUB を流れる。仮想 HUB に実装されている ACL に基づくパケットフィルタや、ACL よりも複雑なルールを設定できるセキュリティポリシー機能が、すべての VPN プロトコルの通信に対して適用できる。仮想 HUB を流れる MAC フレームは、単一のログモジュールによってディスクに書き出すことができる。

ネットワーク管理者や、マルチテナント仮想ホスティング機能を利用する場合の各顧客は、VPN プロトコルを意識せずに共通化されたインターフェイスを用いて SoftEther VPN Server の管理を行うことができる。利用可能なインターフェイスは、以下の 3 種類である。

3.4.4.1　GUI

VPN Server Manager と呼ばれるグラフィカルユーザインターフェイス (GUI) を実装した。Windows および WINE 環境のインストールされた Linux 上で動作する。図 3.6 および図 3.7 は、GUI の画面例を示す。

3.4.4.2　CUI

vpncmd と呼ばれるコマンドラインユーザインターフェイス (CUI) を実装した。SoftEther VPN Server のサポートするすべての OS で動作する。

3.4.4.3　RPC

SoftEther VPN Server は Remote Procedure Call (RPC) を受け付ける。マルチテナント仮想ホスティングサービス等で、管理用ポータルサイトや顧客管理システムから SoftEther VPN Server を制御したい場合に利用する。この場合、、これらのシステムに

図 3.6: SoftEther VPN Server で複数の仮想 HUB を管理する画面

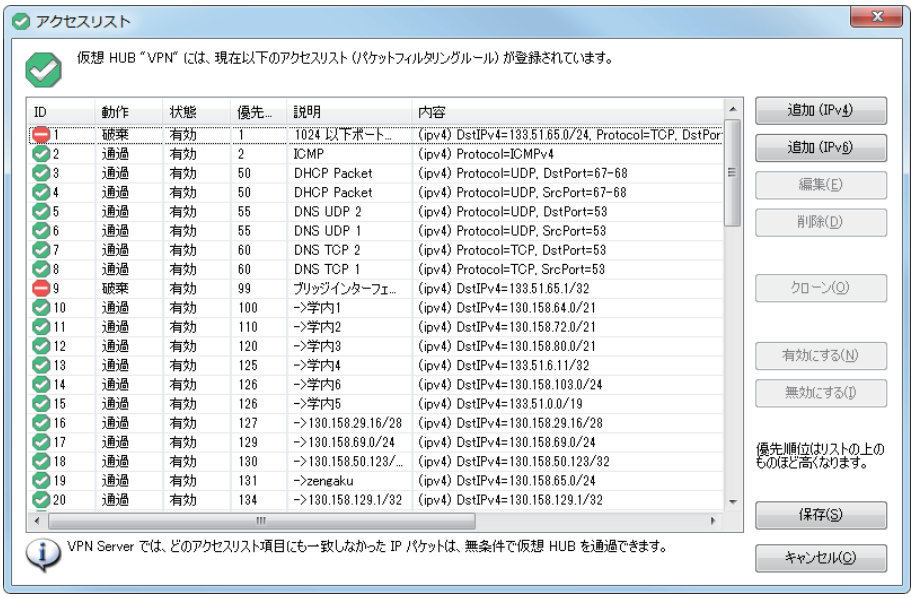


図 3.7: SoftEther VPN Server の仮想 HUB のアクセスリストを設定する画面

RPC クライアントを実装して SoftEther VPN Server に RPC 接続する。GUI および CUI ツールは、内部的にはこの機能を呼び出している。

本研究では、SoftEther VPN Server に、135 個の RPC のプロシージャを実装した。実装したプロシージャの一覧を、以下に機能別にまとめる。

VPN サーバの管理関係

GetCaps, GetServerInfo, GetServerStatus, SetServerPassword, SetServerCert, GetServerCert, GetServerCipher, SetServerCipher, SetKeep, GetKeep, GetConfig, SetConfig, SetSysLog, GetSysLog, RebootServer, Crash, GetAdminMsg, Flush, Debug

VPN プロトコルモジュール (指定ポートでの待受け) の有効化、無効化、設定変更等

CreateListener, EnumListener, DeleteListener, EnableListener, SetIPsecServices, GetIPsecServices, AddEtherIpId, GetEtherIpId, DeleteEtherIpId, EnumEtherIpId, SetOpenVpnSstpConfig, GetOpenVpnSstpConfig, MakeOpenVpnConfigFile, SetSpecialListener, GetSpecialListener

ロードバランシングの管理

SetFarmSetting, GetFarmSetting, GetFarmInfo, EnumFarmMember, GetFarmConnectionStatus

VPN サーバに内蔵されているダイナミック DNS クライアント機能等の管理

GetDDnsClientStatus, ChangeDDnsClientHostname, GetAzureStatus, SetAzureStatus, GetDDnsInternetSetting, SetDDnsInternetSetting

VPN セッション等の管理 (VPN サーバ全体)

EnumConnection, DisconnectConnection, GetConnectionInfo

仮想 HUB の作成、削除、設定、列挙

CreateHub, SetHub, GetHub, EnumHub, DeleteHub, SetHubOnline, GetHubStatus, SetHubLog, GetHubLog, GetDefaultHubAdminOptions, GetHubAdminOptions, SetHubAdminOptions, GetHubExtOptions, SetHubExtOptions, SetAcList, GetAcList, SetHubMsg, GetHubMsg

仮想 HUB のログ関係の管理

EnumLogFile, ReadLogFile

仮想 HUB のユーザ認証設定

GetHubRadius, SetHubRadius

リモートの仮想 HUB 間を接続する機能の管理

CreateLink, EnumLink, GetLinkStatus, SetLinkOnline, SetLinkOffline, DeleteLink, RenameLink, GetLink, SetLink

仮想レイヤ 3 スイッチ (仮想 HUB 間に作成できる仮想的な IP ルータ) の管理

AddL3Switch, DelL3Switch, EnumL3Switch, StartL3Switch, StopL3Switch, AddL3If, DelL3If, EnumL3If, AddL3Table, DelL3Table, EnumL3Table

仮想 HUB の ACL (パケットフィルタ) の設定

AddAccess, DeleteAccess, EnumAccess, SetAccessList

仮想 HUB のユーザ認証機能の設定

CreateUser, SetUser, GetUser, DeleteUser, EnumUser, CreateGroup, SetGroup, GetGroup, DeleteGroup, EnumGroup

PKI 認証関係

AddCa, EnumCa, GetCa, DeleteCa, EnumCrl, AddCrl, DelCrl, GetCrl, SetCrl, RegenerateServerCert

仮想 HUB に接続されている VPN セッションの管理

EnumSession, GetSessionStatus, DeleteSession

仮想 HUB の MAC アドレステーブルなどの内部状態の管理

EnumMacTable, DeleteMacTable, EnumIpTable, DeleteIpTable

ユーザモード NAT 機能および関連機能の管理

EnableSecureNAT, DisableSecureNAT, SetSecureNATOption, GetSecureNATOption, EnumNAT, EnumDHCP, GetSecureNATStatus

ブリッジ機能の管理

EnumEthernet, AddLocalBridge, DeleteLocalBridge, EnumLocalBridge, GetBridgeSupport, EnumEthVLan, SetEnableEthVLan

3.4.5 プラットフォーム抽象化レイヤ

SoftEther VPN Server は、VPN サーバ機能を実装する上位のプログラムコードと、それらのプログラムが呼び出す下位のプラットフォーム抽象化レイヤのコードとに分割されている。これにより、VPN サーバ機能を提供するプログラムを 1 回書くだけで、複数の OS 用にプログラムをビルドでき、それぞれのプラットフォーム上で同等に動作させることができる。

プラットフォーム抽象化レイヤは、Windows、Linux、Mac OS X、FreeBSD および

Solaris 用に実装されている。Windows とそれ以外の UNIX 互換 OS とは大きく異なるが、UNIX 互換 OS でもそれぞれ細かな違いがある。

プラットフォーム抽象化レイヤが実装する機能は、以下のような低レベル処理である。

3.4.5.1 メモリ管理

一般的なメモリ管理関数である malloc、realloc、free に加え、OS が提供するより高速な API が利用可能である場合は利用する。例えば、Windows ではデバッグ時にはヒープ破損のチェックが可能な malloc 系を利用するが、リリース時には HeapAlloc 系 API を利用することによりオーバーヘッドを軽減する。

3.4.5.2 スレッドおよび同期

システム間のスレッドモデルの違いを吸収する。Windows では Win32 スレッド、イベント、クリティカルセクションを使用し、UNIX 互換 OS では pthread によるスレッド、Condition Variables、パイプおよび mutex を使用する。これらのオブジェクトの取扱い (特に待機関数の性質や再帰ロックの可否等) の差異を吸収する。また、スレッドプールを実装し、一度作成したスレッドを再利用することで、一部の環境でのライブラリ側におけるスレッド作成時にメモリリークが生じるバグを回避する。

3.4.5.3 ソケット

TCP および UDP 通信を行う際のソケット API は OS によって挙動が異なる。たとえば非同期通信を実現するためには Windows ではオーバーラップ IO、IO 完了ポート (I/O completion ports) およびイベントへの関連付けが利用可能であり、UNIX 互換 OS では select、poll、epoll が利用可能である。UNIX 互換系における差異としては、非同期モードに関係する ioctl のコードが異なったり、すでに受信キューにデータが入っている場合に新たにデータを受信した際の通知の有無が異なったり、送信バッファが空いたのに通知されなかったりするなどの、細かな部分がある。また、connect() や accept() を別スレッドからキャンセルする方法についても違いがある。このような差異を吸収する。

3.4.5.4 文字列処理

wchar_t 型のサイズやフォーマットが異なったり、sprintf 系関数におけるフォーマット書式が異なったりする部分を吸収する。また、文字コード変換において Windows では Native Language Support (NLS) API を、UNIX では iconv を使用する。

3.4.5.5 モノトニック時刻の供給

モノトニック時刻 [52] とは、OS 起動中に常にインクリメントされる、相対的な、逆戻りすることがないことが保証されている時計の値である。システム時計の変更に応じてモノトニック時刻が影響されてはならない。VPN 通信のタイムアウトや再送処理、内部的な仮想 TCP/IP スタックの処理等に多用されている。SoftEther VPN Server で必要な 10ms の精度を、低いオーバーヘッドで得る必要がある。

Windows においては timeGetTime() API で確実に取得できるが、UNIX 互換 OS によっては clock_gettime() を用いても正しく取得できない場合がある。また、取得できてもオーバーヘッドが大きい場合もある。これらの差異を吸収するだけではなく、OS によってはオーバーヘッドを削減するためにモノトニック時刻を 10ms 間隔で取得し続ける専用のスレッドを作成している。

3.4.5.6 デバッグ支援モジュール

開発時においてメモリリークおよびリソースリークを発見するために、独自のオブジェクト管理コードを実装した。メモリ以外にもファイルディスクリプタやソケット、ロック等のリソースの作成、削除を追跡できる。また、リークしている行番号も表示可能であ

表 3.3: SoftEther VPN Server への接続テストに用いた VPN クライアントの一覧

VPN プロトコル	VPN クライアント
L2TP/IPsec	iPhone (iOS 4, 5, 6, 7, 8) iPad (iOS 4, 5, 6, 7, 8) Android (2, 3, 4) Windows Vista, 7, 8, 8.1, RT Mac OS X (10.6, 10.7, 10.8, 10.9)
SSTP	Windows Vista, 7, 8, 8.1, RT
OpenVPN L3	OpenVPN Client 2.2 for Linux OpenVPN Client 2.2 for Windows OpenVPN Connect for iOS OpenVPN Connect for Android
OpenVPN L2	OpenVPN Client 2.2 for Linux OpenVPN Client 2.2 for Windows OpenVPN Connect for iOS OpenVPN Connect for Android
L2TPv3/IPsec	Cisco 892J, Cisco 1812J, IIJ SEIL x86
EtherIP	NEC IX2015
SEVP	SoftEther VPN Client for Windows SoftEther VPN Client for Linux

る。デッドロックが発生した場合は、ロックの参照元を辿り循環参照を発見することもできる。これらのデバッグ支援機能は、リリース時には自動的に削除され、高速なコードが生成される。

3.4.6 SoftEther VPN Client

SoftEther VPN Server がサポートしている VPN プロトコルのうち、SEVP (SoftEther VPN Protocol) に対応した VPN クライアントとして、SoftEther VPN Client がある。SoftEther VPN Client は、本研究における SoftEther VPN Server と同時に実装した。SoftEther VPN Client は、カーネルモードドライバとして動作する仮想ネットワークアダプタ、VPN 処理モジュールおよび GUI から構成される。SoftEther VPN Client は、SoftEther VPN Server との間で、SEVP プロトコルに従い、Ethernet を SSL にカプセル化して、TCP 上で伝送する。TCP を確立した後、UDP が利用可能であれば、Ethernet over UDP トンネルも確立し、できるだけ UDP を用いて Ethernet を伝送する。SoftEther VPN Client は、Windows および Linux 上で動作する。

3.5 評価

本節では、SoftEther VPN Server について定性的および定量的な側面から評価を行う。定性的な評価においては、複数の VPN プロトコル間における相互運用性およびマルチテナント機能について評価を行う。定量的な評価においては、パフォーマンスの計測および SoftEther VPN Server の普及度合いを示す。これらの結果により、SoftEther VPN Server が 3.2 項において述べた問題を最小のオーバーヘッドで解決できることを示す。さらに、SoftEther VPN Server を実運用環境に導入した結果について評価を行う。これにより、SoftEther VPN Server が既存の VPN サーバと同等の安定性を有していることを示す。

3.5.1 相互運用性

表 3.3 は SoftEther VPN Server をテストする際に用いた VPN クライアントの一覧である。これらのすべての VPN クライアントは、SoftEther VPN Server に接続することができた。

表 3.4 は、広く利用されている VPN サーバおよびそれらの VPN サーバがサポートしている VPN プロトコルの一覧である。SoftEther VPN Server は、PPTP および IPsec トンネリングモードを除き、本表にあるすべての VPN プロトコルをサポートしている。

SoftEther VPN においてこれら 2 個の VPN プロトコルをサポートしなかった理由は、3.3.1 項で述べたように、これらのプロトコルには脆弱性および互換性の問題があり、また、サポートの除外によって、サポートする VPN クライアントの種類が減少しないためである。これらの結果は、3.2.2 項で述べた、複数の VPN プロトコルのサポートのために複数の VPN サーバを同時稼働させる場合に生じる以下の問題を解決できたことを意味する。

1. VPN サーバの管理が複雑となる問題。
2. 複数のハードウェアリソースが必要となる問題。

3.5.2 マルチテナント仮想ホスティング

3.2.3 項において、VPN サーバはマルチテナント仮想ホスティングが必要である旨を述べた。また、3.3.6 項において、SoftEther VPN Server におけるマルチテナント仮想ホスティングの実装について述べた。

本研究では、単一の VPN サーバインスタンスにおいて複数の仮想 HUB の作成、およびタグ付き VLAN のサポートを実装した。これにより、サービスプロバイダは顧客のために VPN サーバを運用することができる。したがって、SoftEther VPN Server はマルチテナント仮想ホスティングをサポートしている。

3.5.3 必要なモジュールのプログラムへの内包

3.2.4 項において、VPN サーバが複数の外部プログラムに依存することにより発生する問題について述べた。また、3.3.7 項において、この問題を、1 つのプログラムへの各モジュールの内包により解決する手法を述べた。

表 3.4: 広く利用されている VPN サーバおよび各 VPN サーバがサポートしている VPN プロトコルの一覧

	レイヤ 3(IP パケット) を伝送する VPN プロトコル					レイヤ 2(MAC フレーム) を伝送する VPN プロトコル			
VPN サーバ	IPsec トンネルモード	L2TP / IPsec	PPTP	SSTP	OpenVPN L3	OpenVPN L2	L2TPv3 / IPsec	EtherIP / IPsec	SEVP
Microsoft RRAS		✓	✓	✓					
Mac OX X Server		✓	✓						
OpenVPN					✓	✓			
企業向けルータ	✓	✓	✓				✓		
PacketiX VPN 3.0									✓
SoftEther VPN Server		✓		✓	✓	✓	✓	✓	✓

企業向けルータとは、Cisco、Extreme Networks、Brocade、F5 Networks 等の販売するアプライアンスを意味する。

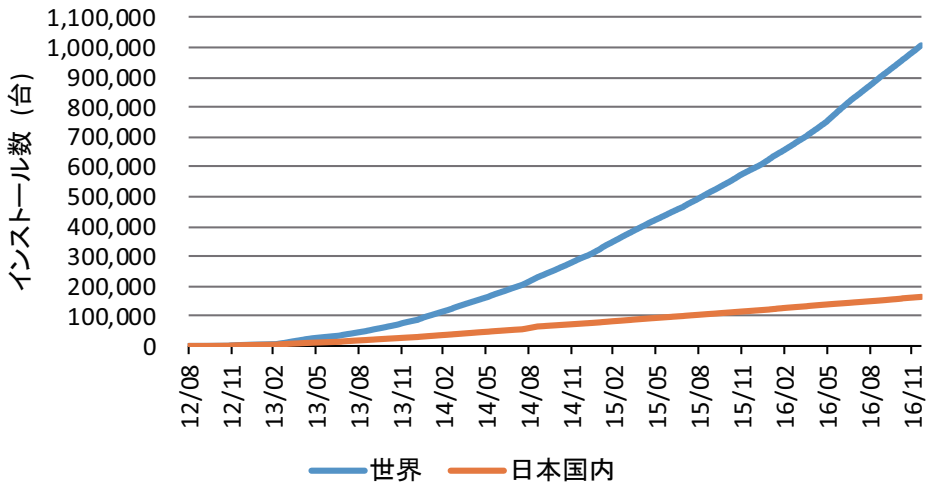


図 3.8: SoftEther VPN Server のインストール数累計

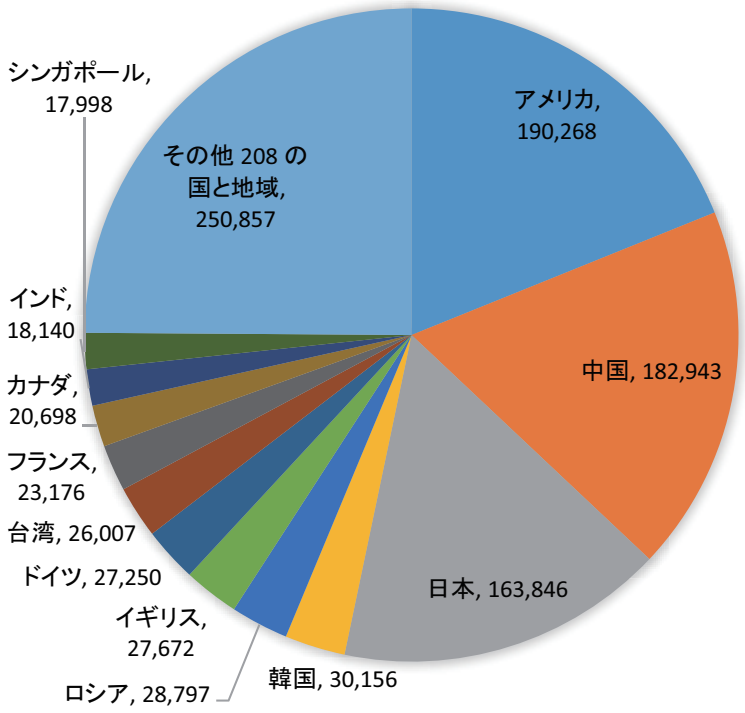


図 3.9: SoftEther VPN Server がインストールされたサーバの所在国とインストール数

本研究で実装したSoftEther VPN Serverは、スレッド作成、スレッド同期処理、メモリ管理、ファイル入出力、ソケットAPIなどの、幅広いOSで利用できるOSの機能以外の外部プログラムに依存しないプログラムとして実装した。Windows、Linux、Mac OS X、FreeBSDおよびSolarisで、VPNプロトコルの処理に必要なPPPやIPsecなどのプロトコル処理や、パケットフィルタなどのセキュリティ処理が、他の外部プログラムを起動することなく、SoftEther VPN Serverのプログラムのみで処理できるようになった。ただし、パケットフィルタ処理は、VPNを流れるパケットに対してのみ適用される。VPNサーバを動作させているコンピュータで、VPNとは無関係の他のアプリケーションやサーバに対するパケットフィルタの設定を行いたい場合は、従来どおり、OSのパケットフィルタが必要である。

これにより、SoftEther VPN Server本体のインストールや、SoftEther VPN Serverを拡張して実装したVPNアプリケーションのインストール処理は、プログラムファイルの実行形式を対象システムにコピーするか、コンパイル済みのオブジェクトファイルを対象システム上でリンクするだけで完了することが可能となった[*5]。システム管理者は、従来のVPNサーバソフトウェアのように、外部依存プログラムの一覧を調査し、これらのプログラムをインストールしてから、VPNサーバソフトウェアを実行する必要がなくなった。

SoftEther VPN Serverを拡張して実装したVPNアプリケーションのインストール処理が簡単となったことを用いたVPNアプリケーションの例については、第4章で述べる。

3.5.4 SoftEther VPN Serverの公開およびインストール実績

本研究では、SoftEther VPN Serverのバイナリパッケージを2013年3月に公開[*6]し、2014年1月にソースコードをGNU General Public License (GPL) Version 2としてGitHubで公開[*7]した。

SoftEther VPN Serverは、2016年12月20日までに、世界中で1,009,000回インストールされた。インストール数は、SoftEther VPN Serverに組み込まれておりデフォルトで有効化されているダイナミックDNS (DDNS) クライアントプログラムのDDNSホスト登録数によって推計した。インストール数の推移は、図3.8のとおりである。図3.9は、SoftEther VPN Serverがインストールされたサーバのipアドレスによって分析した所在国の分布を示す。日本での利用は約16 %であり、残りの84 %は日本以外の国 (ア

[*5] OS側の問題により、あるバージョンのOS用にリンクした実行可能ファイルが別のバージョンで正しく動作しないことがある。特にLinuxではこの問題が顕著である。

[*6] http://www.softether.org/

[*7] https://github.com/SoftEtherVPN/SoftEtherVPN/

メリカ、中国、ヨーロッパ、その他) である。

3.5.5　不具合の改良や機能の拡張

SoftEther VPN Server は、各モジュールが 1 つの実行可能ファイルに統合されているため、2.2.3 項で述べた Visual Studio などと比較して、1 つの実行可能ファイルのみをコピーすれば動作するという利点がある。また、特定のモジュールのみを更新することはできない。このことは、2.1.2 節で述べた Dependency Hell を解決でき、修正版のプログラムのインストールが容易であるという利点がある。一方で、一部のモジュールのファイルのみの差し替えができないため、プログラマは、一部の改良モジュールのみを作成することはできない。プログラムの改良を行いたいプログラマは、SoftEther VPN Server のソースコード全体にアクセスできる必要がある。

オープンソース化は、この問題を解決するための 1 つの方法である。本研究では、SoftEther VPN Server をオープンソースとして公開し、多数のプログラマが、一部のモジュールの不具合を修正したり、機能を改良したりできるようにした。2014 年 1 月に SoftEther VPN Server のソースコードを GitHub で公開した後、2016 年 11 月末までに、著者以外の 23 人のコントリビュータから Pull Request (コードを改善したり、機能を強化したりするためのパッチの送付) を受け、これらのコードを SoftEther VPN Server にコミットした。これらのコントリビュートされたコードの中には、軽微なバグ修正以外にも、対応アーキテクチャを拡張するもの、対応 OS を増加させるもの、SSL の設定のオプションを新設してセキュリティを強化するもの、CPU 使用率を削減するもの、RADIUS 認証における機能を拡張するもの、セキュリティ機能の 1 つにおいて Classless Inter-Domain Routing (CIDR) に対応するもの、などの VPN サーバの機能面、性能面を強化するパッチが多数含まれている。また、SoftEther VPN Server に、Internet Key Exchange Version 2 (IKEv2) という新たな VPN プロトコルのサポートを追加しようとするコントリビュータがおり、メールで実装方法に関する助言を求める質問を受けたので、回答を行った。ただし、2016 年 12 月時点では、そのコントリビュータによる IKEv2 に対応するコード実装は完了していない。これらの結果は、SoftEther VPN Server のソースコードが、多くのプログラマによって改良されることができる程度の保守性、拡張性を有していることを示す。今後も継続的に第三者によってプログラムが改良されることが期待できる。

Dependency Hell を解決し、かつ複数のモジュールを異なるファイルとして分離しておく別の方法として、複数プロセスに分離し、Remote Procedure Call (RPC) などでプロセス間通信を行う方法がある。複数プロセスに分割する方法の利点は、実行時に一部のモ

ジュールを差し替えることができる点にある。SoftEther VPN Server は、シングルプロセスで実装されているため、実行時のモジュールの差し替えができない。しかし、VPN サーバのプロセスを一度終了して実行可能ファイルを差し替え、再起動するためにかかる時間はわずかであるため、実用上はこのことはそれほど問題にならない。また、オーバヘッドは、プロセス内の関数呼び出しのほうが、プロセス間の RPC の呼び出しよりも少ない。VPN サーバの処理においては、モジュール間のデータのやりとりが多く発生する。たとえば、仮想 HUB モジュールと、L2 アダプタモジュールと、VPN プロトコルのモジュール間では、パケットの送受信の度に受渡しが発生する。パケットの受渡しのときに、データ本体は共有メモリを用いて渡し、RPC やプロセス間のイベント通知などで共有メモリへの書き込みが完了したことを通知する方法も考えられる。単一プロセス内のスレッド間でメモリを共有する仕組みは OS 間で同一である一方、異なるプロセス間の共有メモリの実現方法は OS 間で異なるため、OS への依存が高まる。したがって、VPN サーバでは、モジュール間を RPC で接続する方法は適していないと考えられる。

3.5.6 パフォーマンス

本項では、SoftEther VPN Server のパフォーマンスについて述べる。特に、SoftEther VPN Server が複数の VPN プロトコルの通信を同時に処理する際のオーバーヘッドが小さく、通信速度が従来の VPN サーバと比較して高速である点について述べる。

3.5.6.1 実験環境およびベンチマークプログラム

実験においては、3 台の同一仕様の PC を使用した。各 PC は Intel Xeon E3-1230 3.2GHz CPU、16GB のメモリおよび 1 枚の 10 Gigabit Ethernet NIC (Intel 10 Gigabit CX4 Dual Port Server Adapter、2 個の CX4 ポートを有する) を有する。PC の OS は Windows Server 2008 R2 x64 を基本として使用し、実験の必要上 Linux が必要な場合 (OpenVPN を用いた実験を行う場合) においては Linux Kernel 2.6.32 (x64) を使用した。

速度測定には、Windows および Linux で動作する、SoftEther VPN の付属ユーティリティである TrafficServer / TrafficClient ツールを使用した。本ツールは iperf [32] と同様に、一定期間内にできる限り多くのパケットを送受信し、スループットを測定する。iperf と異なり、本ツールは同時に 32 本の TCP コネクションを確立し、そのうち 16 本は送信方向、残り 16 本は受信方向でスループットを同時に測定可能である。本ツールのソースコードは SoftEther VPN Server のソースコードの一部として公開されている。

図 3.10 に、3 台の実験用 PC の物理的な接続を示す。VPN 通信速度を計測する前に、

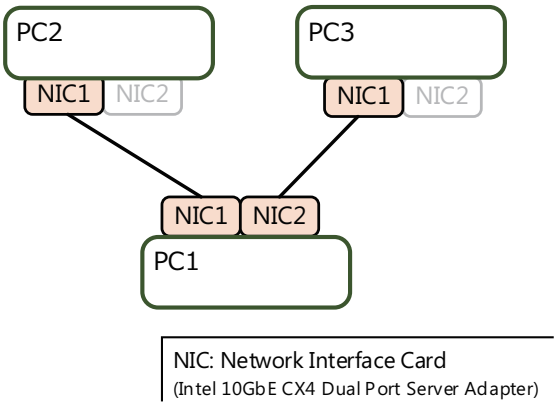
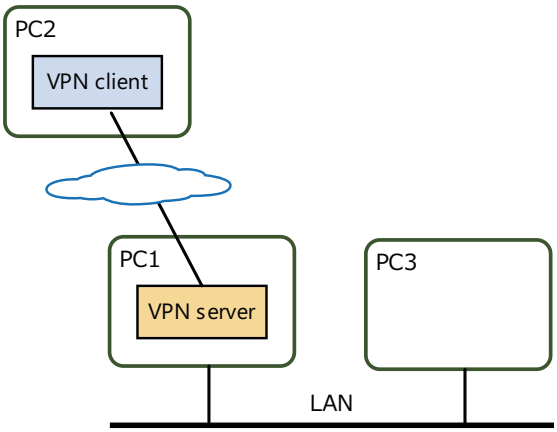
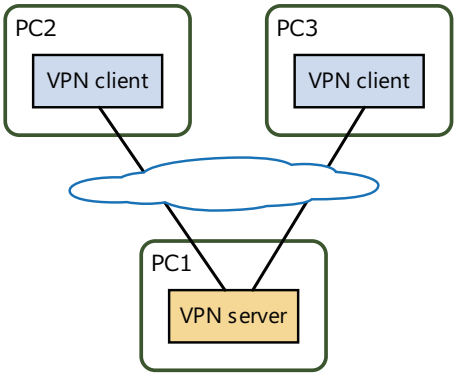


図 3.10: 3 台の実験用 PC の物理的な接続図



(a) VPN クライアントと LAN 上の PC との間の VPN 通信



(b) 2 台の VPN クライアント同士の通信

図 3.11: 実験環境における論理的接続図

まず、PC1 をルータとして、PC2 と PC3 との間のスループットを測定した。スループットは約 9.95Gbps であった。これにより、PC1 の CPU は 10Gbps の接続を十分利用できる程度に高速であることを確認した。以下の実験では、CPU によって処理される暗号化のためにスループットはより低速となっている。

3.5.6.2 VPN プロトコル

本研究では、表 3.5 にある各 VPN プロトコルについて、SoftEther VPN Server と既存の VPN サーバプログラムとの間でのパフォーマンスの比較を行った。SoftEther VPN Server は実際には表 3.4 にある VPN プロトコルすべてに対応しているが、これらの中のいくつかの VPN プロトコルには、ハードウェアルータが必要となるなど、Windows および Linux 環境において動作させることが困難なものがある。そこで、これらのプロトコルは速度測定から除外し、表 3.5 のプロトコルのみ測定を行った。

OpenVPN を含むテストにおいては、OpenVPN L3 プロトコルを用いた。また、当初は Windows 環境において実験を行ったが、OpenVPN Technologies, Inc. によって配布されている OpenVPN の Windows 版プログラムにおける UDP データグラムの扱いが最適化されておらず、そのプログラムがボトルネックとなったために 100Mbps 以上の速度が出なかったため、代わりに Linux 環境で実験を行った。

L2TP を含むテストにおいては、OpenVPN との組み合わせ実験の際にはサーバ、クライアント共に xl2tpd 1.3.6 (Linux 2.6.32 上) を用い、それ以外の実験の際には Windows Server 2008 R2 組み込みのソフトウェアを用いた。

OpenVPN との組み合わせ実験は、ネイティブ Linux 上で動作が可能な L2TP/IPsec および SEVP のみを対象とした。Linux 上で動作する SSTP サーバの実装は SoftEther VPN Server 以外には存在しなかったため、SSTP と OpenVPN との組み合わせは行っていない。

3.5.6.3 実験の構成

スループット測定実験は、図 3.11 に示すように 2 種類の構成で行った。

表 3.5: 性能測定に利用した VPN プロトコル、既存のネイティブ VPN サーバおよび VPN クライアント

VPN プロトコル	暗号化アルゴリズム	比較対象の VPN サーバ	VPN クライアント
L2TP	IPsec (AES128-CBC)	Microsoft RRAS (Windows Server 2008 R2 上)	Windows Server 2008 R2 組み込み
L2TP	IPsec (AES128-CBC)	xl2tpd 1.3.6 (Linux 2.6.32 上)	xl2tpd 1.3.6 (Linux 2.6.32 上)
SSTP	RC4	Microsoft RRAS (Windows Server 2008 R2 上)	Windows Server 2008 R2 組み込み
OpenVPN (L3)	AES128-CBC	OpenVPN 2.2.2 (Linux 2.6.32 上)	OpenVPN 2.2.2
SEVP	RC4	SoftEther VPN Server	SoftEther VPN Server

1. 実験 A. 1 台の VPN クライアントと LAN 上の 1 台の PC との VPN 通信 (図 3.11 (a))。
   PC1 は VPN サーバとして動作し、PC2 は VPN クライアントとして動作する。PC3 では VPN ソフトウェアは動作させず、PC1 上のソフトウェアブリッジを経由して、PC1 に VPN 接続している PC2 と通信を行う。したがって、各 VPN プロトコル単体での性能を測定することができる。この構成は、社内 LAN へのリモートアクセスでよく見られる構成である。
2. 実験 B. 2 台の VPN クライアント同士の通信 (図 3.11 (b))。
   PC1 は VPN サーバとして動作し、PC2 および PC3 は VPN クライアントとして動作する。この構成においては、2 個の VPN クライアントは、実験 B1 においては同一、実験 B2 においては異なる VPN プロトコルを使用して VPN サーバに接続する。したがって、異なる VPN プロトコル同士の各種組み合わせにおける性能を測定することができる。

3.5.6.4 実験結果

図 3.12 に実験 A の結果を示す。これは、単一の VPN プロトコルを用いて、VPN クライアントと LAN 上の PC との VPN 通信を実施したものである。L2TP においては、SoftEther VPN Server は Windows 標準の RRAS 機能とほぼ同等の速度となった。SSTP においては、Windows 標準の RRAS 機能のほうが SoftEther VPN Server よりも高速な結果となった。これは、RRAS の VPN モジュールがすべて Windows カーネルに実装されているのと比較して、SoftEther VPN Server の大部分はユーザプロセスとして実装されていることが理由である。SSTP および SEVP は L2TP よりも高速となった。これは、SSL において軽量なアルゴリズムである RC4 が使用されているためである。OpenVPN においては、SoftEther VPN Server よりも OpenVPN のオリジナル実装のほうが約 12% 高速な結果となった。この結果は、SoftEther VPN Server の実装と OpenVPN のオリジナル実装の複数の相違点による複合的な原因によるものであると考えられる。たとえば、SoftEther VPN Server には、プログラムを簡単にするために、スレッドを多用している部分がある。VPN セッションごとにスレッドが 1 つ作成されるため、スレッド間のコンテキストスイッチによる影響が大きい。

図 3.13 に実験 B1 の結果を示す。これは、同一の VPN プロトコルを用いて、2 台の VPN クライアント間で VPN サーバを経由した通信を実施したものである。L2TP、SSTP および SEVP においては実験 A と同等の結果となったが、OpenVPN においては SoftEther

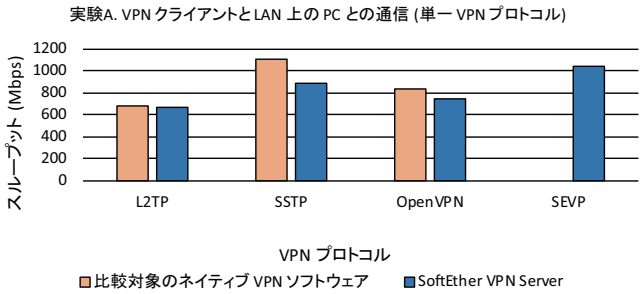


図 3.12: 1 台の VPN クライアントと LAN 上の 1 台の PC との VPN 通信スループット

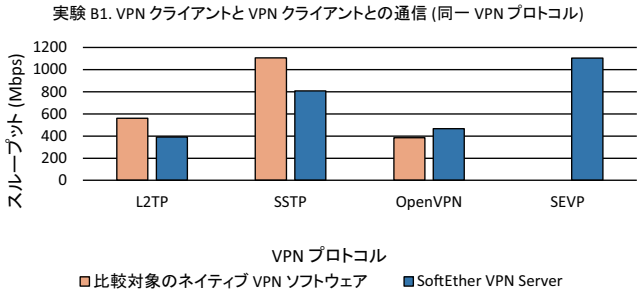


図 3.13: 同一 VPN プロトコルを用いた 2 台の VPN クライアント同士の VPN 通信スループット

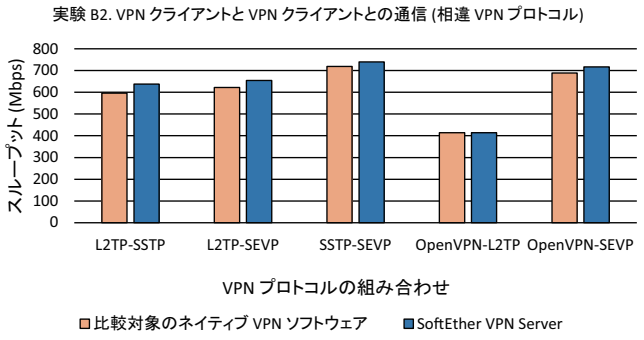


図 3.14: 異なる VPN プロトコルを用いた 2 台の VPN クライアント同士の VPN 通信スループット

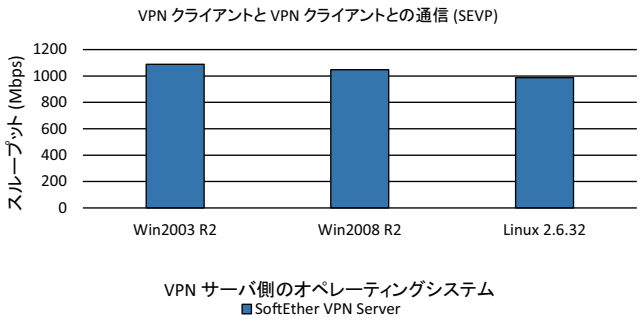


図 3.15: 異なる OS において SoftEther VPN Server を動作させた場合の VPN 通信スループット

VPN Server のほうが高速となった。その理由として、実験 B1 では VPN サーバは 2 本の VPN コネクションを同時に処理しなければならず、その際に AES の復号化と暗号化の処理を行わなければならないが、OpenVPN はシングルスレッドで複数の VPN セッションを処理しているため AES の処理を同一のプロセッサで実行しなければならず、一方、SoftEther VPN は複数スレッドで複数の VPN セッションを処理しているため 2 個の AES 処理が並列して動作するため高速な結果となっている点が挙げられる。

図 3.14 に実験 B2 の結果を示す。これは、異なる VPN プロトコルを用いて、2 台の VPN クライアント間で VPN サーバを経由した通信を実施したものである。実験 B1 の結果と異なり、すべての組み合わせについて SoftEther VPN Server のみを使用した場合が、比較対象の 2 つのネイティブ VPN ソフトウェアを使用した場合と、少なくとも同じ程度の速度か、または高速な速度を実現していた。L2TP および SSTP の組み合わせの場合については、SoftEther VPN Server は RRAS と比較して 7% も高速であった。高速な結果となった理由は、2 個の VPN サーバを組み合わせて使用する場合と比較して、SoftEther VPN Server の単一インスタンスで処理を行う方式のほうがメッセージコピーおよびコンテキストスイッチの回数が減少するためである。

SoftEther VPN Server は、OS 間の移植性を有する。Windows だけではなく、Linux、Mac OS X、FreeBSD および Solaris でも動作する。図 3.15 は、特に Windows Server 2003 R2 x64、Windows Server 2008 R2 x64 および Linux カーネル 2.6.32 を動作させた CentOS 6 において同一条件で VPN 通信を行った際のスループットの比較である。この実験においては、実験 A と同様に SEVP を用いて実験を行った。古いバージョンの Windows Server が最も高速な結果となった。Linux においては Windows よりも 6% 低速であり、さらなる最適化の余地が残されている。

3.5.7 安定性

SoftEther VPN Server は、以下のような多様な環境で安定して動作することから、十分な安定性を有している。

3.5.7.1 VPN Gate Server への組み込み

本研究では、SoftEther VPN Server を用いて VPN Gate [78] システムを実装した。VPN Gate は、Great Firewall of China 等の検閲用ファイアウォールを回避するための中継システムである。VPN Gate システムにおけるボランティアの Windows PC 上では、VPN Gate Server と呼ばれる中継サーバが稼働する。VPN Gate Server は SoftEther

VPN Server を組み込んだプログラムである。VPN Gate Server は、本研究において実装した L2TP、SSTP、OpenVPN L3 および SEVP のクライアントを受け付ける。VPN Gate について、詳しくは第 4 章で述べる。

本研究では、VPN Gate を 2013 年 3 月に公開した。2016 年 12 月時点では、常時約 10,000 台のボランティアサーバが世界中で稼働している。このうち 32 台は、本研究でホストしているサーバである。全世界のボランティアサーバには、2013 年 3 月から 2014 年 9 月までに合計 7.8 億本の VPN コネクションが接続され、12.7 ペタバイトのトラフィックが中継された。このうち、本研究でホストする 32 台のサーバは、合計 4,469 万本の VPN コネクションを受け付け、2.5 ペタバイトのトラフィックを中継した。これらの期間中に 1 件のクラッシュにつながるバグが発見された。当該バグは、SoftEther VPN Server とは無関係の、VPN Gate Server への拡張時に記述されたコードに関係するバグであった。当該バグ以外に、VPN Gate Server のクラッシュや継続的なメモリリークは発見されていない。

3.5.7.2 仮想 HUB レンタルサービスの実験

本研究では、SoftEther VPN Server のマルチテナント仮想ホスティング機能を評価するため、8 台の SoftEther VPN Server をインストールしたサーバをインターネットに接続し、誰でも Web サイトから仮想 HUB を作成し登録することができるようにした。本仮想 HUB レンタルサービスは、2006 年 10 月に、SoftEther VPN Server の元となった商用版 PacketiX VPN Server を用いて開始された。2014 年 9 月までに作成された仮想 HUB の数は合計 141,270 個であり、合計 4 ペタバイトのトラフィックを転送した。仮想 HUB は、ユーザ登録により自動的に作成され、2 ヶ月以上 1 度も利用されない場合は自動的に削除される仕組みである。これには、3.4.4 で述べた RPC が利用されている。また、8 台のサーバは 3.3.2 項で述べたフォールトトレランスおよびロードバランス機能を使用してグループ化し、ユーザからは 1 個のシステムに見えている。

2014 年 9 月時点では、グループ全体で常時約 2,300 個の仮想 HUB をホストしており、グループ全体で約 3,000 本の VPN セッションが確立されている。平均的にみると、約 5 割の仮想 HUB には VPN セッションが接続されておらず、約 1 割の仮想 HUB には VPN セッションが 1 本接続されており、約 2 割の仮想 HUB には VPN セッションが 2 本接続されており、約 1 割の仮想 HUB には VPN セッションが 3 本接続されており、残りの約 1 割の仮想 HUB には VPN セッションが 4 本から 50 本程度まで接続されている。VPN セッションが 1 本だけ常時接続されている仮想 HUB が約 1 割ある。これは、ユーザ拠点

へのリモートアクセス VPN サーバとして利用するために、ユーザ拠点側で常時稼働している VPN サーバからの VPN セッションが常時接続されているものである。そして、モバイル環境等からユーザが必要に応じて同一の仮想 HUB に VPN 接続している間のみ、セッション数が 2 以上となる。したがって、1 台のサーバには平均約 375 本の VPN セッションが接続されている。

本仮想 HUB レンタルサービスは、2006 年 10 月から現在まで安定して稼働している。サービス開始当初はデッドロックが発生する問題が 1 件、サーバが再起動した後に全 VPN クライアントから大量の SSL 通信が同時に接続されようとして CPU 負荷が高くなり、全 VPN クライアントの接続処理でタイムアウトが生じて再接続が繰り返される問題が 1 件発見された。デッドロックの問題については、ロック関係の実行時のグラフを分析して解決した。また、SSL 通信の大量接続時の問題については、SSL ネゴシエーションフェーズに進入することを許可する同時コネクション数の上限数を設定することにより解決した。当該バグ以外に、本サーバシステムのクラッシュや継続的なメモリリークは発見されていない。

3.5.7.3　製品版ソフトウェア等の実績

SoftEther VPN Server の元となった商用版 PacketiX VPN Server の最初のバージョンである Version 2.0 は、ソフトイーサ株式会社より 2005 年 12 月に発売された。IPv6 への対応等が実施された次のバージョンである Version 3.0 は、2010 年 3 月に発売された。SoftEther VPN Server によって実装された複数 VPN プロトコル対応バージョンは、Version 4.0 として 2013 年 7 月に発売された[*8]。これらの製品版を導入した企業等の顧客数は、2016 年 12 月時点で約 6,050 である。これらの企業ユーザからは、PacketiX VPN Server のプログラムが原因でクラッシュが発生する報告はない。同様に、3.5.4 項で述べたオープンソース版のユーザからも、SoftEther VPN Server が原因でクラッシュが発生する報告は、パーソナルファイアウォールソフトウェアのカーネルモードモジュールの不具合であると考えられる問題を除き、寄せられていない。

3.5.7.4　アプライアンスへの組み込みの実績

商用版 PacketiX VPN Server 4.0 は、複数のハードウェアベンダから VPN アプライアンスとして出荷されている[*9]。これらのハードウェアは、異なる CPU で動作している

[*8] http://www.softether.jp/1-product/11-vpn/
[*9] http://openblocks.plathome.co.jp/products/eb/packetix/,
http://www.bias.jp/product/packetix_on_bias/

Linux アプライアンスである。ハードウェアの出荷数は非公開であるが、おおむね数百程度である。これらのハードウェア上の PacketiX VPN Server における安定性に関する問題の報告は寄せられていない。

3.5.8　商用 ISP の VPN サービスへの採用

商用版 PacketiX VPN Server は、複数の商用 Internet Service Provider (ISP) において、顧客向け VPN 接続サービスを提供する電気通信サービスの VPN サーバとして採用されている[*10]。一部のサービスでは、顧客からの申込みや解約、VPN 内における IP アドレスの割り当て等に際して PacketiX VPN Server に対して投入するリクエストを自動化している。自動化には、3.4.4 項で述べた RPC が利用されている。このような自動化プログラムは、ISP がサービスを構築する際に ISP 側において開発された。

これらの商用 ISP サービスにおいては、PacketiX VPN Server 側が原因の不具合としては 1 件のみ発生した。それは、アクセスリストとして N 個のエントリが登録されている場合、大量の IP ブロードキャストパケットが仮想 HUB を流れる際に N^2 回分のアクセスリストエントリに対する走査が行われるという問題に起因していた。あるサービスでは仮想 HUB に約 1,000 個のアクセスリストエントリが登録されていたため、ブロードキャストパケットを処理するパフォーマンスが低下し、スループットの低下と遅延の増加が生じた。この問題を解決し、ブロードキャストパケットであっても N 回分のアクセスリストエントリの走査で処理が完結するようにした。この問題以外に PacketiX VPN Server 側の不具合が原因の問題は発生していない。

3.5.9　評価のまとめ

本研究の目標である統合的な VPN サーバソフトウェアとは、多様な VPN プロトコルを、差異を意識する必要なく同時に利用でき、また、他の特定の OS やライブラリへの依存度が低く、1 つのインスタンスとして動作するような VPN サーバソフトウェアである。

本研究では、本研究の提案手法である L2 アダプタを用いて SoftEther VPN Server を実装した。これにより、レイヤ 2 の VPN プロトコルとレイヤ 3 の VPN プロトコルとの間を接続できるようになった。また、仮想レイヤ 2 スイッチを共通のバスとして使用し、サポートするすべての VPN プロトコルを統一的に処理することができるようにした。ただし、本実装においては、レイヤ 2 のプロトコルは Ethernet に限っている。Ethernet 以

[*10] http://www.interlink.or.jp/service/myip/,
http://vpn.kozukata.co.jp/

外を対象としたレイヤ 2 VPN プロトコルは、本実装ではサポートしていない。しかし、3.3.1 項で述べたように、Ethernet のみのサポートで実用上問題ない。

1.1 項で述べたように、従来の VPN サーバは、VPN プロトコルごとに 1 つずつ異なる VPN サーバプロセスを実行する必要があった。例えば、1.1 項の図 1.1 のように、VPN サーバの管理者は、L2TP/IPsec、OpenVPN (L3) および EtherIP の 3 つの VPN プロトコルをサポートする VPN サーバをインストールし、管理するには、3 回の操作が必要であった。本研究で実装した SoftEther VPN Server は、複数の VPN プロトコルのためのモジュールを 1 つのプロセスに統合した。これにより、管理者は、統合的な VPN サーバソフトウェアを 1 つインストールするだけで、7 種類のプロトコルをサポートし、幅広く使用されている多様な VPN クライアントソフトウェアや VPN ルータからの VPN 接続を実現することができるようになった。また、ユーザ認証、セキュリティ設定の適用、ログの管理、IP アドレスの割り当て管理などの管理業務を行う際に、1.1 項の図 1.2 のように、1 回の操作だけで、すべての VPN プロトコルの通信にその設定が適用されるようになった。

複数の VPN プロトコル間の通信速度は、従来の VPN サーバソフトウェアを組み合わせた場合よりも、SoftEther VPN Server で複数の VPN プロトコルを処理したほうが、最大約 7 % も高速な結果となり、オーバヘッドが小さくなることを示すことができた。

さらに、SoftEther VPN Server では、従来の VPN サーバでは OS の持つネットワークモジュールに依存していた IP スタックや PPP モジュールなどを、プログラム内に実装した。これにより、マルチテナント仮想ホスティングが実現できるようになった。従来の VPN サーバで、VPN から物理的なネットワーク上のホストへの通信を行う際に OS 上の特権を必要とする操作が必要であった問題も、ユーザモード NAT によって解決し、ユーザ権限のみを用いてリモートアクセス VPN サーバが実現できるようになった。これには、ユーザ権限で利用可能な TCP および UDP ソケットが使用されている。ただし、Raw socket を作成する権限が必要な ICMP 用のソケットは、一部の OS では特権を必要とするため、ユーザ権限のみでは ICMP を用いた外部ホストとの通信 (ping, traceroute 等) ができない。また、ユーザモード NAT は、ブリッジと異なり、VPN と物理的なネットワークとの通信が非対称であるという制限事項がある。

その他にも、IPsec 処理やパケットフィルタ処理などの、従来の VPN サーバソフトウェアを用いる場合に外部プログラムを実行する必要があった処理や、本研究の提案手法である L2 アダプタなどの処理も、VPN サーバソフトウェアのプログラムに内包した。これにより、SoftEther VPN Server は、スレッド作成、スレッド同期処理、メモリ管理、ファ

イル入出力、ソケット API などの、幅広い OS で利用できる OS の機能以外に、外部プログラムに依存しなくなった。インストールが簡単になり、システム管理者は、これらの機能を有する OS 環境上にプログラムファイルを設置すれば VPN サーバを利用できるようになった。ただし、SoftEther VPN Server に統合されたパケットフィルタは、VPN を流れるパケットに対してのみ適用される。

3.6 本章のまとめ

この章では、第 1 章で述べた提案手法に基づいた統合的な VPN サーバソフトウェアである SoftEther VPN Server に関する設計と実装について述べた。SoftEther VPN Server は、単一の VPN サーバインスタンスで複数の VPN プロトコルを統一的にサポートする。サポートされる VPN プロトコルは、L2TP over IPsec、SSTP、OpenVPN L3 および L2、EtherIP over IPsec、L2TPv3 over IPsec およびネイティブの SoftEther VPN Protocol (SEVP) である。これらのプロトコルは、PC、スマートフォンおよび VPN ルータを含む幅広い VPN クライアントをカバーする。複数の VPN プロトコルを単一の VPN サーバによってサポートすることにより、管理者は VPN サーバの設定や管理が容易に行えるようになる。また、マルチテナント仮想ホスティングを実現する。

SoftEther VPN Server の実装における鍵となる特徴は、レイヤ 2 およびレイヤ 3 プロトコルの両方を経由して接続された VPN コネクション間のメッセージを相互に交換するためのソフトウェアベースの仮想 HUB にある。仮想 HUB に、セキュリティ設定や、ログ保存処理が集約されている。この仕組みにより、管理者は集中化された操作によりアクセス制御やログの管理を行うことができる。VPN クライアントがレイヤ 2 フレームを送信したときは、そのフレームはそのまま仮想 HUB に渡される。VPN クライアントがレイヤ 3 パケットを送信したときは、L2 アダプタと呼ばれるモジュールがレイヤ 3 パケットをレイヤ 2 フレームに変換し、仮想 HUB に渡す。L2 アダプタは、一般的な OS における ARP 処理および NIC における MAC フレーム送受信処理に相当する機能を有する。

本研究の実験結果は、本研究の解決策による複数 VPN プロトコルのサポートの際のオーバーヘッドが最小であることを示している。単一の VPN プロトコルを処理する場合においては、SoftEther VPN Server のパフォーマンスは Microsoft Windows の RRAS 機能と同等程度または低速な結果となった。一方で、2 個の VPN クライアントが異なる VPN プロトコルを用いて通信を行う場合においては、SoftEther VPN Server を単体で用いた場合が、従来の VPN サーバプログラムを複数用いた場合と比較して高速な結果となった。

本研究では、SoftEther VPN Server のバイナリパッケージを 2013 年 3 月にフリーウェアとして公開した。また、2014 年 1 月にソースコードを GNU General Public License (GPL) Version 2 として公開した。SoftEther VPN Server は、2016 年 12 月 20 日までに世界中で 1,009,000 回インストールされた実績を有する。SoftEther VPN Server は、商用ソフトウェアである PacketiX VPN Server 3.0 の後継版である。PacketiX VPN Server の新たなバージョン 4.0 は、SoftEther VPN Server で実装したマルチプロトコルのサポート機能のコードが搭載されている。PacketiX VPN Server は 6,050 社に導入されている。

これらの結果により、本研究の目的である、統合的な VPN サーバソフトウェアを実現するために必要な目標のうち、レイヤ 2 とレイヤ 3 の間の統一、および、異なる VPN プロトコル間のセキュリティや設定管理の統一を実現することができた。

今後は、SoftEther VPN Server の他のプラットフォームへの移植およびさらなる最適化を目指す。また、SoftEther VPN Server の持つロードバランシングおよびフォールトトレランス機能の性能について評価を行う。L2 アダプタおよびユーザモード NAT の IPv6 対応は、リモートアクセス VPN において IPv6 が普及するに際して、実現を目指す。SEVP のネイティブ VPN クライアントである SoftEther VPN Client を iOS および Android に移植することも目指す。さらに、多数の拠点間を SoftEther VPN Server で接続する場合における遅延をさらに小さくし、単一障害点を無くすために、通常 Multiple Protocol Label Switching (MPLS) 上で利用されている Virtual Private LAN Service (VPLS) と似た Ethernet over IP のフルメッシュネットワークを構築する機能の実現を目指す。これを実現するために、対応するレイヤ 2 VPN プロトコルとして、VXLAN を追加することを目指す。

第 4 章

統合的な VPN サーバを用いた検閲回避 VPN 中継システム VPN Gate の設計と実装

第 3 章では、統合的な VPN サーバソフトウェア SoftEther VPN Server の設計と実装について述べた。

第 4 章では、実装した統合的な VPN サーバソフトウェアを用いて構築した検閲回避システムである VPN Gate の設計および実装について述べる。第 4 章では、新たに提案する検閲回避技術についても述べる。

4.1 概要

一部の国の政府は、インターネット上に検閲用ファイアウォールを設置し、それらの国の内側から海外のインターネット上の特定のサーバへのアクセスを規制している。例えば、中国の Great Firewall (GFW) は Twitter、Facebook および YouTube へのアクセスを禁止している [9]。これらの検閲が実施されている国のインターネットユーザは、検閲用ファイアウォールを回避するため、公開中継サーバを利用することがある。公開中継サーバとしては、公開プロキシ、公開 VPN サーバおよび Tor ノード [18] などがある。通常、これらの公開中継サーバの IP アドレスのリストは、ユーザに知ってもらう必要があるため、公開の場所に掲載されている。したがって、検閲当局はこれらの IP アドレスを入手し、検閲用ファイアウォールの遮断リストにこれらの IP アドレスを追加することにより、インターネットユーザによる公開中継サーバへのアクセスを遮断できる。さらに、例えば、中国政府はリスト化されていない Tor ノードをスキャンし、自動的に遮断をすることを行っている [104]。Tor の中継サーバは、このような検閲当局によるスキャン活動に対する耐性を有していない [50]。

本研究では、GFW のような検閲用ファイアウォールに対する遮断耐性 (Blocking Resistance) を有する分散型公開 VPN 中継システムを構築する。このシステムを、VPN Gate と呼ぶ [78]。遮断耐性を実現するために、VPN Gate は、ボランティアによって提供される頻繁に変化する多数の IP アドレスを利用する。稼働中の VPN サーバのリストをサーバリストと呼ぶ。このサーバのリストを管理するために、VPN Gate リストサーバを設置する。VPN Gate のユーザは、VPN Gate リストサーバから、サーバリストの一部分のみを取得することができる。そして、取得したサーバリストの中にあるいずれかの VPN サーバに VPN 接続を行うことができる。ユーザは、VPN 接続が確立されている間は、通常は規制されているようなインターネット上のサーバに対して通信を行うことができる。このような仕組みにより、検閲当局が VPN Gate の VPN 中継サーバすべてを遮断することを困難にする。

検閲回避システムを実現する上では、遮断耐性を実現することが重要である。本研究では、2 つの手法を用いて、遮断耐性を実現する。2 つの手法とは、無実の IP アドレスの混入手法および協調的なスパイ発見手法である。1 つ目の無実の IP アドレスの混入手法とは、VPN Gate と無関係な複数の IP アドレスをサーバリストに含める手法である。例えば、極めて重要なサーバ (例: Windows Update のサーバ) を無実の IP アドレスとしてサーバリストに混入する。この手法により、検閲当局は、サーバリスト内のすべての IP アドレスを検閲用ファイアウォールの遮断リストに登録する前に、混入された無実の IP アドレスを識別して取り除く必要が生じる。2 つ目の協調的なスパイ発見手法とは、特定の IP アドレスが無実の IP アドレスであるか否かを識別するための調査用通信を発する検閲当局のコンピュータ (スパイコンピュータ) を検出する手法である。この手法においては、すべてのボランティアの VPN 中継サーバが協調動作することにより、スパイコンピュータであると疑われる IP アドレスのリストを生成し、共有する。このリストを、スパイリストと呼ぶ。各 VPN 中継サーバは、スパイリストに含まれる IP アドレスからのパケットを無視するようになる。その結果、検閲当局は、サーバリストに含まれるある IP アドレスが、本物の VPN 中継サーバであるか、または混入された無実の IP アドレスであるかを見分けることができなくなる。

VPN Gate のシステムは、多数の VPN Gate Server ソフトウェアのインスタンスおよび中央のリストサーバから構成される。また、VPN Gate のユーザのための VPN Gate Client ソフトウェアも利用可能である。ただし、ユーザは、VPN Gate Client ソフトウェアがなくても、OS に標準搭載されている VPN クライアントソフトウェアを用いて、VPN Gate Server に接続することができる。ボランティアは、簡単に VPN Gate Server をイン

ストールし稼働させることができる。特に、ボランティアは TCP や UDP の外向きポートを開放するために Network Address Translation (NAT) [23] 装置の設定を変更する必要がない。ユーザは、VPN Gate Server に Secure Sockets Layer (SSL) [6]-VPN (SSL-VPN) プロトコルを用いて接続することができる。その他に、L2TP/IPsec、OpenVPN および MS-SSTP などの、OS に標準搭載されているか、またはアプリケーションとして追加することができる VPN クライアントがサポートする多様な VPN プロトコルを用いて接続することができる。本研究では、中央のリストサーバを立ち上げ、ボランティアの VPN 中継サーバからの登録を受け付け、サーバリストを生成し、ユーザに対して配布を行った。

本研究では、VPN Gate を 2013 年 3 月 8 日に公開した。同年 8 月 29 日には、VPN Gate サーバの台数は約 3,000 台に達した。この台数は、当時の Tor 中継ノードの数に匹敵する。同日時点で、ユーザからは、すべての VPN Gate サーバに対して、1 日間で合計 464,000 回の VPN 接続が行われた。これらの VPN 接続は、88,000 個のユニークな接続元 IP アドレスからのものであった。

VPN Gate は、GFW に対する遮断耐性を有している。本システムを公開した後、間髪を入れず、GFW は VPN Gate のボランティアの VPN 中継サーバの IP アドレスを GFW の遮断リストに登録し始めた。2013 年 4 月 4 日は、すべてのボランティアの VPN 中継サーバのうち 81% が遮断されてしまい、中国国内からは、残りの 19% のサーバにしかアクセスできなくなった。そこで、無実の IP アドレスの混入手法および協調的スパイ発見手法を実装した。その結果、4 月 26 日には中国国内からの 50% の到達性を実現することができ、5 月 9 日には 75% の到達性を実現することができた。さらに、ボランティアの VPN 中継サーバのうち 40% の IP アドレスは毎日のように頻繁に変化する。これにより、GFW はある時点でのすべてのボランティアの VPN 中継サーバの IP アドレスが入手できたとしても、新たな IP アドレスの出現に追い付くことができなくなる。VPN Gate は、このようにして、中国のインターネットユーザからの安定的な到達性を実現している。2013 年 8 月末の時点で、中国からの VPN 接続の数は 45,000 回であり、これらは 9,000 個のユニークな IP アドレスを接続元とするものであった。これは、中国における Tor ユーザの数である 3,000 ユーザよりも多い。

VPN Gate は、検閲回避システムであり、匿名化システムではない。Tor と異なり、VPN Gate のボランティアの VPN 中継サーバは、パケットログを保存する。また、VPN Gate は匿名化のためのマルチホップ中継機能を有しない。

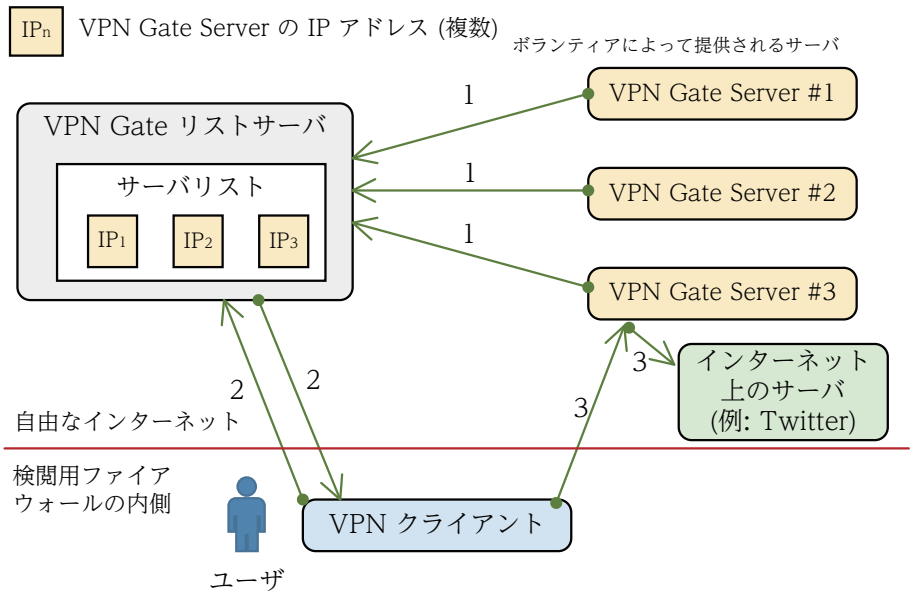


図 4.1: VPN Gate の仕組み

4.2 VPN Gate の仕組み

VPN Gate の仕組みを図 4.1 に示す。ボランティアは、VPN Gate Server ソフトウェアをダウンロードし、自分の PC 上で動作させる。VPN Gate Server は、稼働中は、自分自身を VPN Gate リストサーバに登録する。リストサーバは、稼働中の VPN Gate Server のインスタンスのリストであるサーバリストを管理する。

以下、検閲用ファイアウォールの内側の VPN Gate ユーザが、通常ではアクセスすることができない Web サーバにアクセスするために、VPN Gate を利用する方法を述べる。ユーザは、まず VPN Gate リストサーバの Web サイトにアクセスし、稼働中の VPN Gate Server のリストを取得する。検閲当局が少ないコストですべての稼働中の VPN Gate Server のリストを取得することを防止するため、リストサーバは、サーバリストの一部のみを返却する。次に、ユーザはこの一部のリスト内から任意の 1 台の VPN Gate Server を選ぶ。最後に、ユーザは自分が選択した VPN Gate Server に対して VPN 接続する。この際に、PC の OS に標準搭載されている VPN クライアントを利用する方法と、VPN Gate Client と呼ばれる VPN Gate 専用の VPN クライアントを利用する方法とがある。VPN 接続が確立された後は、VPN Gate Server が、ユーザとインターネットとの

間のすべての通信を中継する。これにより、ユーザは規制されている Web サイトにアクセスすることができるようになる。

4.2.1 ボランティアによる VPN Gate Server の稼働

上述したように、VPN Gate のシステムは、ボランティアが VPN Gate Server を PC 上にインストールして稼働することに依存している。新たにボランティアとなる際には、氏名、住所または個人情報を提供することは要求されない。VPN Gate Server の稼働中は、VPN Gate Server は多数のユーザからの VPN 接続を待受けする。VPN Gate Server は、4 種類の VPN プロトコル (L2TP/IPsec、OpenVPN、MS-SSTP および SoftEther VPN プロトコル) を受け付ける。

VPN Gate Server は、稼働中は、定期的に PC のインターネット接続種別を検出する。PC が NAT 装置の内側に設置されていることを検出した場合、VPN Gate Server は Universal Plug and Play (UPnP) プロトコルまたは UDP ホールパンチング手法を用いて、NAT におけるポート開放を試みる。VPN Gate Server は、インターネット接続種別の検出が完了した後に、4.2.3 項で述べるように、VPN Gate リストサーバに自分自身を登録する。

4.2.2 ユーザによる VPN Gate Server への接続

VPN Gate のユーザは、VPN Gate リストサーバの Web サイトにアクセスし、サーバリストの一部分を取得する。このサーバリストの内容には、ボランティアの VPN 中継サーバの IP アドレス、ポート番号、国・地域、帯域幅や遅延などの回線品質の情報、現在接続されている VPN 接続数、および累計の VPN 接続数などの情報が含まれる。ユーザは、リストの中からこれらの情報を参考にして、好きな VPN 中継サーバを 1 つ選択する。

検閲用ファイアウォールが設置されている国においては、検閲当局が、VPN Gate リストサーバの Web サイトをすぐさま発見し、ファイアウォールの遮断リストにこの Web サイトを登録してしまうことが想定される。すると、その国のユーザはリストサーバの Web サイトにアクセスすることができなくなる。そのような場合は、ユーザは多数の稼働中の VPN Gate Server の有する HTTP [29] 中継機能を用いて、リストサーバの Web サイトにアクセスすることができる。この詳細なメカニズムについては、4.3.5 項で述べる。

次に、ユーザは以下の方法のうち 1 つを選択して、VPN 接続を確立する。

4.2.2.1 オペレーティングシステム (OS) に標準搭載されている VPN クライアントを使用して接続する方法

ユーザは、サーバリスト上から選択した VPN 中継サーバの IP アドレスを、L2TP/IPsec や MS-SSTP VPN クライアントの設定画面にコピーアンドペーストなどの手法により入力する。この設定画面においては、ユーザは VPN 接続を行う際のユーザ名とパスワードを入力する必要があるが、ここではユーザ名を ”vpn”、パスワードを ”vpn” とすればよい。このように、ユーザ名とパスワードは固定であり、ユーザ登録は必要ない。この方法の利点は、ユーザが一切のソフトウェアをインストールする必要がない点にある。

4.2.2.2 OpenVPN クライアントを使用して接続する方法

ユーザは、最初の 1 回のみ、OpenVPN クライアントソフトウェアをダウンロードしてインストールする。その後、VPN 接続を行う都度、VPN Gate リストサーバが OpenVPN 接続設定ファイル (.ovpn ファイル) をダウンロードし、この設定ファイルを用いて OpenVPN クライアントを起動して、宛先の VPN 中継サーバに接続を行う。

4.2.2.3 VPN Gate Client を使用して接続する方法

ユーザは、最初の 1 回のみ、VPN Gate Client ソフトウェアをダウンロードしてインストールする。その後、VPN 接続を行う都度、VPN Gate Client を起動し、表示されるサーバリストから接続先の VPN 中継サーバをクリックする (図 4.2)。すると、指定された VPN 中継サーバへの接続が実行される。この方法の利点は、操作が簡単であること、および 4.3.4 項で述べる間接的サーバリスト転送プロトコルが利用できることである。

4.2.3 VPN Gate リストサーバ

VPN Gate リストサーバのソフトウェアは、稼働中の多数の VPN Gate Server のインスタンスからの登録を受け付け、これらの各サーバのステータスを監視する。VPN Gate リストサーバがユーザからサーバリストを要求された場合は、サーバリスト全体のうち一部分を返却する。VPN Gate リストサーバは、4.3 項で述べるように、検閲用ファイアウォールに対する遮断耐性を有する。

4.3 検閲用ファイアウォール耐性を実現するシステム

VPN Gate は、検閲用ファイアウォール耐性を実現するシステム (Firewall Resistance System) を有する。検閲用ファイアウォール耐性を実現するシステムとは、検閲用ファイ

図 4.2: VPN Gate Client のスクリーンショット

アウォールがユーザの VPN Gate の利用を遮断することに対する耐性を実現するためのシステムである。本システムは、VPN Gate Server および VPN Gate リストサーバの両方に実装されている。この節では、まず、中国の検閲当局が用いている遮断手法について概説する。次に、本研究における検閲用ファイアウォール耐性の実現のための 2 つの重要な手法である、無実の IP アドレスの混入手法および協調的スパイ発見手法について述べる。

4.3.1 中国の Great Firewall において使用されている遮断手法

本研究における 1 つの目標は、中国の GFW に対する遮断耐性を実現することにある。そのためには、まず、GFW の有する遮断手法について研究をする必要がある。既に報告されているところによると [1] [13] [107] [60]、GFW は中国国内のインターネットサービスプロバイダ (ISP) と海外の ISP との境界線上に設置されており、遮断リストに記載されている宛先に対する通信を、以下の 3 つの手法により遮断する能力を有する。GFW を運用する検閲当局は、毎日のように、遮断リストの IP アドレス一覧をメンテナンスしている。遮断リストのメンテナンスのために、人的リソースや自動化されたスキャンシステ

ムを用いていると考えられる。例えば、GFW 管理局は、リスト化されていない Tor ノードに対するスキャンを常時行っていることが判明している [104]。

4.3.1.1 DNS レスポンスの偽造と送信元 IP アドレス偽装による送信

GFW は、国内のインターネットユーザが海外サーバに対して送信するすべての DNS クエリを盗聴している [57]。GFW は、国内の DNS クライアントが、海外の任意の DNS サーバに対して、遮断リストに記載されているドメイン名を含む Fully Qualified Domain Name (FQDN) に関する DNS クエリを送信したことを検出すると、直ちに偽造された DNS レスポンスを DNS クライアントに対して送信する (DNS レスポンスパケットの送信元 IP アドレスも偽装される)。この際、GFW は DNS クエリパケットを遮断することはしないので、DNS クエリパケットは宛先の本物の DNS サーバに届き、本物の DNS サーバからの正規のレスポンスも DNS クライアントに届く。しかし、DNS クライアントは偽物と本物のレスポンスを区別することができず、先に届いたレスポンスの結果を採用する。このようにして、GFW は、パケットを遮断することなく、クライアントに対して遮断リストに登録されているドメイン名を含む FQDN の名前解決を妨害することができ、結果的に、クライアントは宛先サーバにアクセスすることができなくなる。この遮断手法は、DNS に依存しない通信に対しては利用できない。

4.3.1.2 TCP RST パケットの偽造と送信元 IP アドレス偽装による送信

GFW は、国内のインターネットユーザが海外サーバに対して送信するすべての TCP パケットを盗聴している。GFW は、国内の TCP クライアントが、遮断リストに登録されているホスト宛に TCP 接続を確立しようとするか、または TCP のペイロードに遮断リストに登録されている URL やキーワードが含まれていることを検出すると、直ちに偽造された TCP RST パケットを TCP クライアントに対して送信する (TCP RST パケットの送信元 IP アドレスも偽装される) [103]。GFW は、この手法を用いて、共有 Web サーバにおける一部の URL へのアクセスのみを遮断することができる。この遮断手法は、暗号化されたペイロードや、TCP 以外の通信 (UDP など) に対しては利用できない。

4.3.1.3 IP アドレスの遮断

GFW は、国内のインターネットユーザから海外に送信されようとする IP パケットのうち、宛先 IP アドレスが遮断リストに含まれている IP アドレスである場合は、その IP パケットを遮断することができる。この手法を実現するために、GFW は国内と海外との間を接続する国際 ISP に対して、偽の BGP ルーティングテーブルを広報し、パケットを

吸い寄せる手法を用いるか、または、すべてのパケットが通過する境界上にパケットフィルタ設備を設置している。この遮断手法は、暗号化された TCP パケット、UDP パケット、および DNS に依存しない通信を遮断することができるが、宛先 IP アドレス単位での遮断しか行うことができない。

4.3.2 無実の IP アドレスの混入手法

VPN Gate において、検閲用ファイアウォールに対する遮断耐性を実現するための 1 つ目の手法として、無実の IP アドレスの混入手法を提案する。本手法の概要を、図 4.3 に示す。本手法においては、VPN Gate リストサーバが VPN 中継サーバのリストをユーザに返却する際に、無実の IP アドレスと呼ばれる、偽の VPN 中継サーバの IP アドレスを複数個含めるようにする。無実の IP アドレスには、VPN Gate と無関係の IP アドレスが選ばれる。本手法の効果を高めるために、無実の IP アドレスには、インターネット上で極めて重要な IP アドレスを含めることが望ましい。無実の IP アドレスとしてとても良い例としては、DNS ルートサーバやトップレベルドメインの DNS 権威サーバの IP アドレス、Windows Update サーバの IP アドレス、および多くのユーザが使用しているメールサーバの IP アドレスなどがある。検閲当局は、無実の IP アドレスが混入されていることにすぐに気付くであろう。そうすると、検閲当局は、サーバリストに含まれる IP アドレスを自動的に検閲用ファイアウォールの遮断リストに追加することを躊躇するようになる。検閲当局は、各 IP アドレスが本物の VPN Gate Server の IP アドレスであるかどうかを検証してから、その IP アドレスを遮断リストに追加する必要が生じる。なお、本手法を用いる際には、無実の IP アドレスを常時混入しておく必要はない。検閲当局が、「常に VPN Gate のサーバリストには突然無実の IP アドレスが混入されることがあるから、サーバリストを無条件に信用することはできないので、遮断の際には、十分注意しなければならない。」と警戒をする程度に、少量の無実の IP アドレスを、時々混入しておけばよい。

本研究では、無実の IP アドレスの混入手法を実施する際には、免責のため、以下のような警告文を掲載することにした。

> VPN Gate リストサーバの Web サイトに、VPN サーバリストには誤った IP アドレスが含まれていることがあります。このサーバリストを検閲用ファイアウォールに投入すると、意図しない不具合が生じる場合があります。そのため、検閲用ファイアウォールの IP 遮断リストの管理のために、VPN サーバリストを使用してはな

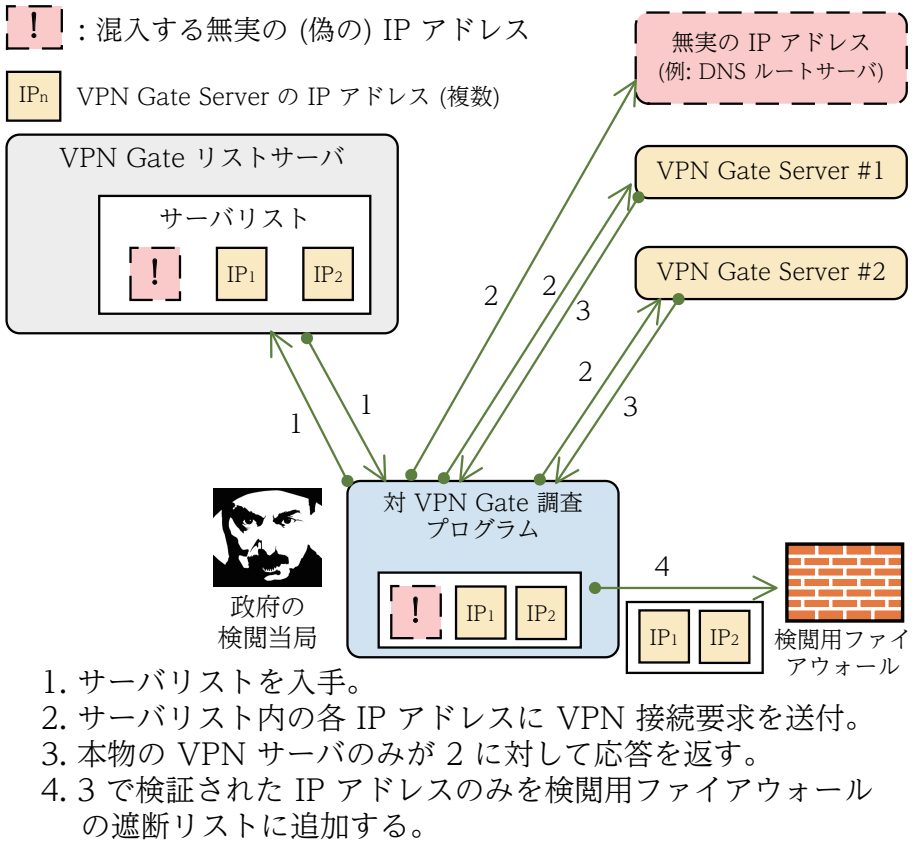


図 4.3: 無実の IP アドレスの混入手法

りません。

無実の IP アドレスの混入手法は、VPN Gate の一般的なユーザに影響を与えない。もし、ユーザが無実の IP アドレスを VPN 接続先として選択した場合でも、単に、接続エラーが発生するだけである。その場合、ユーザはサーバリストにある他の IP アドレスを選択すればよい。

無実の IP アドレスの混入手法は、無実の IP アドレスとして指定されたサーバに対して、分散サービス拒否攻撃 (DDoS) [70] を生じさせることはない。例えば、1 日あたり 1 億ユーザが VPN Gate を使用したと仮定する。本研究では、1,000 台の VPN 中継サーバあたり 1 個の無実の IP アドレスを混入する。もし、各ユーザがサーバリストから接続先の VPN 中継サーバを無作為に選択したとすると、この無実の IP アドレス宛には、1 日あたり 100,000,000 / 1,000 = 100,000 回の接続要求が到達することになる。1 回の接続要求あたり合計 5 回の接続試行パケットが到達したと過程しても、無実の IP アドレスが受け取る無駄なパケット数は、1 秒あたり 7 パケット以下である。このように、非常に少な

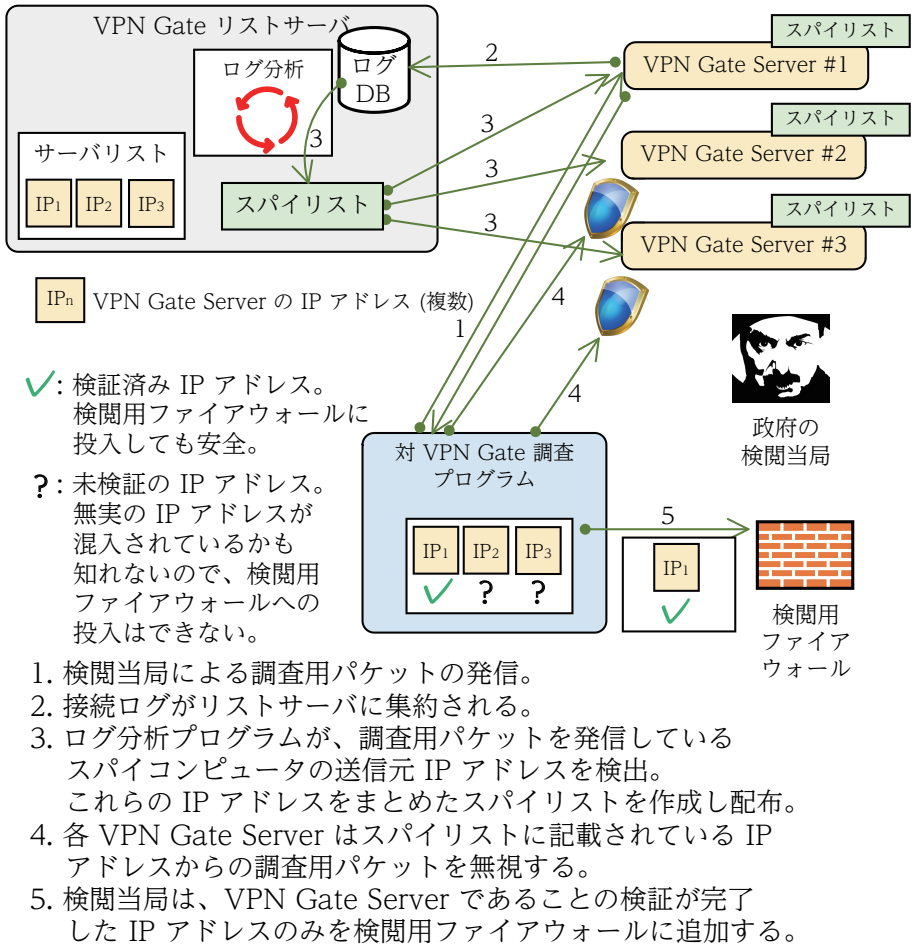
い数の無駄なパケットを受け取ることは、今日のインターネットサーバの処理能力からすると、全く影響がないと考えてよいであろう。

実際には、一般的なユーザは VPN 中継サーバをランダムに選択することはせずに、リストの上のほうから順に接続を試みる場合が多い。例えば、ユーザが 100 台の VPN 中継サーバのリストを取得するとき、そのリストのちょうど中頃に無実の IP アドレスを混入する。ユーザは、リストの上から順に接続を試していき、接続に成功したにそれ以降は接続試行をしないので、ユーザが無実の IP アドレスに接続を試行する頻度は低くなる。これにより、無実の IP アドレスとして混入されたサーバに、無駄なパケットが届く回数は削減される。

4.3.3 協調的なスパイ発見手法

本研究では、VPN Gate における検閲用ファイアウォールに対する遮断耐性を実現するための 2 つ目の手法として、協調的なスパイ発見手法を提案する。この手法は、検閲当局による、サーバリスト内の各 IP アドレスが本物の VPN Gate Server であるかどうかを検証するための調査用パケットを発するコンピュータ (スパイコンピュータ) を発見することを目的とする。スパイコンピュータを発見するために、すべての VPN Gate Server のインスタンスが協調動作し、スパイコンピュータであると疑われる IP アドレスのリスト (スパイリスト) を作成する。図 4.4 に示すように、スパイリストがあれば、各サーバは、スパイリストに含まれる IP アドレスからの調査用パケットを無視することができる。スパイリストは、IP アドレス、および IP アドレスの範囲の集合で構成される。

本手法を実現するには、協調的な動作が必須である。スパイコンピュータが、一般的な VPN プロトコルに準拠した調査用パケットを調査対象となる IP アドレスに送付するとき、その IP アドレスで動作する VPN Gate Server は、それ単体では、そのパケットがスパイコンピュータからの調査用パケットであるか、それとも、一般的なユーザからの接続要求であるか、見分けることができない。1 個の VPN Gate Server のインスタンスだけで、接続元のコンピュータがスパイコンビュータか否かを見分けることができない以上、スパイコンピュータがその VPN Gate Server に対して VPN 接続を成功させてしまったときには、すでにスパイコンピュータによって VPN Gate Server であるとして識別された状態となっており、手遅れである。したがって、VPN サーバは、スパイコンピュータからの最初の接続要求パケットに対して応答を返す以前に、その接続要求がスパイコンピュータからのものであるかどうかを知っていなければならない。それを知ることは、1 個の VPN Gate Server のインスタンス単体では不可能である。



1. 検閲当局による調査用パケットの発信。
2. 接続ログがリストサーバに集約される。
3. ログ分析プログラムが、調査用パケットを発信している
   スパイコンピュータの送信元 IP アドレスを検出。
   これらの IP アドレスをまとめたスパイリストを作成し配布。
4. 各 VPN Gate Server はスパイリストに記載されている IP
   アドレスからの調査用パケットを無視する。
5. 検閲当局は、VPN Gate Server であることの検証が完了
   した IP アドレスのみを検閲用ファイアウォールに追加する。

図 4.4: 協調的スパイ発見手法

この問題を解決するためには、すべての VPN Gate Server が協力をしてスパイコンピュータを発見し、スパイコンピュータの一覧を共有するスパイリストを作成し、スパイリストに含まれるクライアントからの接続要求を無視しなければならない。スパイリストの生成は、下記の 2 段階の処理によって行われる。

4.3.3.1 VPN Gate Server における処理

VPN Gate Server は、VPN 接続要求を受け取った際に、その接続ログを記録する。接続ログは、完全接続記録、不完全接続記録および性急な接続記録の 3 種類に分類される。完全接続記録は、クライアントから VPN 接続要求があった後に、VPN 接続が確立され、その後、実際に閾値を超える時間の接続の維持状態、またはデータ量の伝送が生じた場合に記録される。不完全接続記録は、クライアントから VPN 接続要求があったが、接続過程におけるネゴシエーション中に通信が切断されたり、プロトコルエラーが発生したりし

て、VPN 接続確立に至らなかった場合に記録される。性急な接続記録は、VPN 接続は確立されたものの、閾値を超えるデータ量の伝送を行わずに、閾値の時間以下で切断された場合に記録される。VPN Gate Server は、これらの接続記録に、接続元 IP アドレス、日時、データ転送量および切断までの時間などのメタデータを付加して記録する。各 VPN Gate Server は、接続記録を一定時間ごとに VPN Gate リストサーバに送付する。

4.3.3.2 VPN Gate リストサーバにおける処理

VPN Gate リストサーバは、すべての VPN Gate Server のインスタンスから集約される接続ログを用いて、以下の手順により、スパイコンピュータを発見する。

1. 多数の VPN サーバが、特定の IP アドレスまたは IP アドレス範囲からの不完全接続を記録している場合は、その IP アドレスまたは IP アドレス範囲を、スパイコンピュータとしてマークする。
2. 多数の VPN サーバが、特定の IP アドレスまたは IP アドレス範囲からの性急な接続を記録している場合は、その IP アドレスまたは IP アドレス範囲を、スパイコンピュータとしてマークする。

VPN Gate リストサーバは、上記の処理を定期的に実行して生成されたスパイリストを、すべての VPN サーバに配布する。スパイリストのサイズを縮小するために、近寄った IP アドレスが複数あるときは、これらを集約し、1 つの/24 のサイズのブロックで記載する。この集約手法は、集約の検討対象となる IP アドレスからの接続の頻度やその他の状況を含めて決定される閾値を超えたときに適用される。例えば、中国で登録されている IP アドレスにスパイコンピュータが多く潜んでいることが観測されたならば、スパイリストの精度を犠牲にしてでもリストのサイズを縮小するために、アメリカなど他の国の IP アドレスと比較して、集約開始のための閾値を低くするといった制御を行っている。

4.3.4 サーバリストのユーザへの配布方法

通常、中継サーバを用いる方式の検閲回避システムは、検閲当局に中継サーバのアドレスを知られたり、中継サーバに対する通信を遮断されたりすることなく、クライアントに対して、中継サーバのアドレスを知らせる必要があるという難問に直面する [28]。VPN Gate においては、この問題を解決するために、3 つの手法を組み合わせる。

1 個目の手法として、キースペース・ホッピング [28] という手法を用いる。キースペース・ホッピングとは、各クライアントが、サーバリストの全体のうち、疑似乱数を用いて

生成されたユニークな部分集合のみを取得する手法である。これは、無線通信において、無線ノードが混信を避けるために周波数ホッピング方式を用いることと似ている。キースペース・ホッピングにおいて使用される疑似乱数のシード値として、クライアントのネットワークアドレスの一部を使用する。この方法により、検閲当局は、VPN Gate のサーバリスト全体を取得するために、広い範囲および多くの個数の IP アドレスを必要とし、サーバリスト全体を取得するコストを高めることができる。

2 個目の手法は、間接的サーバリスト転送プロトコルの導入である。政府によるインターネット検閲が実施されている国においては、VPN Gate Client がリストサーバから最新のサーバリストを取得しようとする通信が、検閲用ファイアウォールによって遮断されてしまう可能性が高い。間接的サーバリスト転送プロトコルは、この問題を解決することができる。本プロトコルは、VPN Gate Client が中間サーバを経由してサーバリストを取得することができるようにする。中間サーバは、すでにクライアントによってアドレスが知られている VPN Gate Server のインスタンスである。中間サーバを経由する方式によりクライアントに転送されるサーバリストは、中間サーバによって改ざんされないように、リストサーバによってデジタル署名される。

3 個目の手法は、VPN Gate Client の配布パッケージへの初期サーバリストの動的埋め込みである。ユーザが VPN Gate Client を最初に起動するときには、VPN Gate Client がダウンロードされた時点での初期サーバリストが予め埋め込まれている。VPN Gate Client は、この初期サーバリストを用いて、いずれかの到達可能なサーバ経由で、間接的サーバリスト転送プロトコルを用いて最新のサーバリストをダウンロードできる。本方式を実現するためには、VPN Gate Client がダウンロードされようとする都度、サーバリストを配布パッケージ内に埋め込む処理が必要になる。この際、埋め込むサーバリストの生成には、上記で 1 個目の手法として述べたキースペース・ホッピング手法が利用される。また、このサーバリストにも、無実の IP アドレスが混入されている。

2013 年 8 月 19 日までに、VPN Gate リストサーバは、1,630,000 回のサーバリストのダウンロード要求のうち 23.2% にあたる 379,000 回の間接的サーバリスト転送プロトコルを用いたダウンロードに応答した。

4.3.5　HTTP 中継機能および日替わりミラー URL メール配布サービス

検閲当局にとって、VPN Gate リストサーバの Web サイトや VPN Gate Client のダウンロードサイトを遮断することは容易である。この問題に対して、本研究では、VPN Gate Server に HTTP 中継機能を実装する。この機能は、ユーザが最初に VPN Gate Client

をダウンロードする際に、検閲を回避してダウンロードする機会を提供する。本機能は、VPN Gate Client を使用せずに、OS 組み込みの VPN クライアント機能を使用して VPN Gate を利用するユーザに対して、VPN Gate リストサーバの Web サイトへのアクセス手段を提供する。

検閲用ファイアウォールは、HTTP 中継機能を経由した HTTP 通信であっても、キーワードベースの検閲と遮断を行うことができる。このような検閲を困難にするため、HTTP 中継機能を経由するときには、コンテンツは gzip 圧縮 [17] された HTTP 応答として回答される。

VPN Gate は、日替わりミラー URL メール配布サービスを提供する。本サービスは、HTTP 中継機能を経由して VPN Gate リストサーバの Web サイトにアクセスできる URL の一覧を、毎日ユーザに対してメール配信することにより提供するものである[*1]。これらの URL のリストを受け取ったユーザは、それを自分自身で利用でき、また、URL の一覧を、オンラインまたはオフラインのソーシャルネットワークを用いて拡散することができる。これらの拡散は、インターネット検閲の対象国の中でも可能である。2013 年 9 月の時点で、本サービスは、11,000 人の加入者を有している。加入者が本サービスに登録する際のネットワークアドレスは、キースペース・ホッピングに必要な疑似乱数のシード値として利用される。

4.4 実装

この節では、VPN Gate Server、VPN Gate Client および VPN Gate リストサーバの実装について述べる。

4.4.1 VPN Gate Server の実装

本研究では、VPN Gate Server を Windows におけるアプリケーションソフトウェアとして実装した。本ソフトウェアのコードは、第 3 章で実装した SoftEther VPN Server をベースとした。

SoftEther VPN Server は、L2TP over IPsec、Ether IP over IPsec、OpenVPN L3 および L2、L2TPv3 over IPsec、SSTP および SoftEther VPN プロトコルの 7 種類の VPN

[*1] 中国国内の電子メールサービスは、検閲当局によりメールの内容が検閲されている場合があるため、本サービスにより配信されるメールが遮断される可能性がある。一方、中国国内のユーザは、海外の電子メールサービス、たとえば Gmail を利用することができる。この場合、本サービスのメールは日本から Gmail のサーバに届くので、その時点では GFW に検閲されない。中国国内から Gmail にアクセスする際にも、SSL が利用されるため GFW に検閲されない。この手法は当初有効であったが、GFW 管理局は 2014 年 12 月に中国国内から Gmail サーバへのアクセスをすべて遮断したため、現在は Gmail は利用できない。

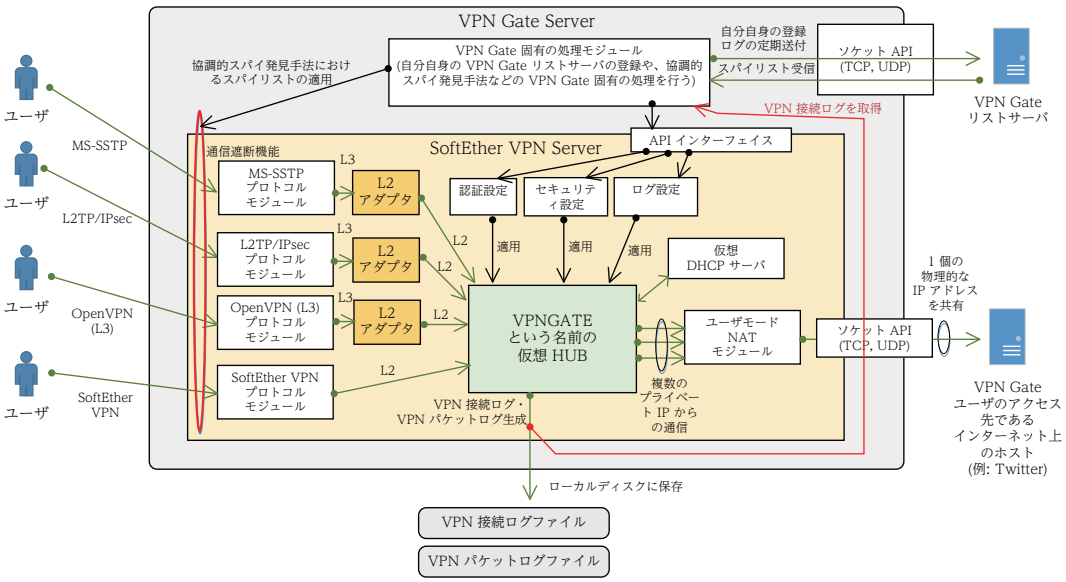


図 4.5: VPN Gate Server の実装

プロトコルをサポートしている。このうち、本研究では、Windows、Mac OS X、iOS および Android が対応している VPN プロトコルである、以下の 4 種類の VPN プロトコルをサポートすることにした。

1. L2TP/IPsec
2. OpenVPN L3
3. MS-SSTP
4. SoftEther VPN プロトコル

図 4.5 は、SoftEther VPN Server を内蔵した VPN Gate Server の構造を示す。VPN Gate Server を立ち上げると、”VPNGATE” という名称の仮想 HUB が、SoftEther VPN Server 内に作成される。VPN Gate Server に対して設定されるセキュリティ設定やユーザ認証などの設定は、これらの各 VPN プロトコルに、統一的に適用される。たとえば、VPNGATE 仮想 HUB には、ユーザ名およびパスワードが ”vpn” であるユーザが 1 つ作成される。VPNGATE 仮想 HUB に接続した ”vpn” ユーザ間では通信が行えないようにしたり、”vpn” ユーザが、ボランティアの PC 本体やボランティアの所属する LAN のプライベート IP アドレスのホストにアクセスできないようなセキュリティ設定が適用される。また、VPN Gate Server が出力するログファイルは、複数の VPN プロトコルが使用されている場合でも、1 つのログに統一され、フォーマットも統合される。VPN 接続ログは、ファイルに保存されるほか、VPN Gate 固有のプログラムにもデータが渡される。

VPN 接続ログは、協調的スパイ発見手法で利用するために、VPN Gate リストサーバに対して定期的に送付される。

VPN Gate Server は、VPN 接続している複数のユーザの間で、インターネット接続を共有する機能を必要とする。1 個のインターネット接続のための IP アドレス (グローバル IP アドレスまたは NAT の内側のプライベート IP アドレス) を、複数の VPN クライアントに割り当てたプライベート IP アドレスで共有する必要がある。そのために、VPN Gate Server では、3.3.5 項で述べたユーザモード NAT を使用している。これにより、ボランティアは、OS のネットワーク設定を変更したり、NAT を実現する他のプログラムをインストールしたりすることなく、VPN Gate のユーザに対して、インターネット接続を提供することができる。また、ボランティアは OS 上での特権を使わずに VPN Gate Server を動作させることができ、他のアプリケーションの通信に支障が発生することもない。

従来の VPN サーバソフトウェアの場合、PPP プログラムや IPsec プログラム、パケットフィルタプログラム等が OS 環境上にインストールされていなければ動作しなかった。SoftEther VPN Server は、3.3.7 項で述べたように、外部プログラムへの依存の解消を実現している。これにより、VPN Gate Server のインストールは、プログラムファイルを OS 環境上にコピーするだけで完了する。これは、ボランティアにとってとても簡単な作業である。

4.4.2 NAT 装置の内側での VPN Gate Server の動作

本研究では、ほとんどのボランティアは NAT 装置の内側で VPN Gate Server を稼働させることを想定した。ボランティアに NAT におけるポート開放の設定をしてもらうことは難しい。多数の到達可能なボランティアサーバを確保するためには、VPN Gate Server は、NAT におけるポート開放の処理も統合的に行う必要がある。そこで、VPN Gate Server には、UPnP プロトコルおよび UDP ホールパンチング手法を用いて、NAT 装置のポートを自動的に開放する機能を実装した。この機能は、4.3.4 項で述べた間接的サーバリスト転送プロトコルのサーバ機能を動作させる際にも利用できる。

4.4.3 稼働中の VPN サーバの状態監視

VPN Gate リストサーバは、すべての登録されている VPN サーバの状態監視を行う。状態監視は、VPN サーバが最初に VPN Gate リストサーバに登録をしてきた際だけではなく、その後も一定時間ごとに実施される。VPN Gate リストサーバは、各 VPN サーバが正常に稼働しているかどうかを検査してから、サーバリストにその VPN サーバが搭載

されるようにする。

VPN Gate リストサーバは、各 VPN サーバの正常動作を検証するだけではなく、各 VPN サーバのインターネット接続回線の品質についても検証し、その結果を集約する。各 VPN サーバのインターネット接続回線の通信遅延の測定のために、Google Public DNS サーバ (8.8.8.8) [37] に対する ICMP [85] エコー要求を送付する。また、帯域幅の測定のために、本研究において設置する速度測定試験サーバ[*2] との間の TCP 速度測定を実施する。これらの測定結果は、VPN サーバによって、VPN Gate リストサーバに定期的に送付される。ユーザは、VPN サーバごとに表示されるこれらの測定結果を参考にして、できるだけ低遅延で、かつ高帯域なサーバを選択し、VPN 接続をすることができる。

4.4.4 VPN 接続ログおよびパケットログ

各 VPN サーバは、VPN クライアントが VPN 接続を確立または切断した記録である VPN 接続ログを保存する。また、各 VPN サーバは、ユーザが VPN 接続を経由して送受信したパケットの記録であるパケットログを保存する。パケットログには、TCP または UDP のヘッダ部分が保存される。また、ペイロードについても保存される。ただし、すべてのヘッダやペイロードが保存されると、ディスク容量を大きく消費するため、特に重要な TCP または UDP ヘッダや、HTTP アクセスが行われた場合の HTTP ヘッダ (TCP のペイロード) のみが保存される。パケットログとして保存される内容は、以下のとおりである。

1. TCP パケット。VPN サーバは、SYN、SYN+ACK および ACK フラグビットを持つ TCP パケットについて、IP ヘッダおよび TCP ヘッダを保存する。また、TCP のペイロードが HTTP リクエストヘッダである場合は、アクセス先の HTTP の URL を保存する。それ以外の場合は、保存をしない。
2. UDP パケット。VPN サーバは、DHCP パケット、IPsec および OpenVPN プロトコルの接続要求パケットについて、IP ヘッダおよび UDP ヘッダを保存する。これらのヘッダを保存する理由は、ユーザはこれらの VPN プロトコルを用いて多段 VPN 接続を行うことができ、これによりユーザの真の IP アドレスを隠す活動を行うことができるためである。これら以外の場合は、保存をしない。

[*2] 速度測定試験サーバは、筑波大学内に接続速度で設置した。これは SINET (Science Information NETwork) を用いてインターネットとの間で 1Gbps の通信が可能である。海外からの通信は、SINET と海外の ISP との間にボトルネックがある場合があるため、本来は世界中の様々な ISP に測定サーバを立ち上げることが望ましい。

それぞれのボランティアは、自分の PC 上で動作している VPN サーバの VPN 接続ログおよびパケットログを確認することができ、接続元の VPN クライアントの IP アドレスを得ることができる。もし、違法行為を目的としたユーザが VPN サーバを利用したとき、そのサーバの所有者は、公的機関に関連するログを提出することができる。VPN サーバは、また、4.3.3 項で述べたように、VPN 接続ログを、協調的スパイ発見手法のために、VPN Gate リストサーバに転送する。この際、パケットログは転送されない。

本研究では、VPN Gate が、匿名化システムとして利用されにくいほうが良いと考えた。そこで、上記で述べたログが意図的に保存されるようにすることにより、ユーザが VPN Gate を用いた違法行為を抑制する。その一方で、一般的なユーザのプライバシーが必要以上に侵害されないようにする必要もある。もしパケットログを保存する機能がなければ、違法行為を目的としたユーザは、VPN Gate を、真の IP アドレスを隠蔽する手段として、盛んに悪用するようになるであろう。パケットログが VPN サーバに保存されていれば、その VPN サーバを運用するボランティアは、法執行機関の求めに応じて、そのパケットログを提出することができる。VPN Gate の Web サイトには、VPN Gate 不正利用防止ポリシーが掲載されており、各 VPN Gate Server は違法行為を目的とするユーザに悪用されることを防ぐためにパケットログが保存されている旨が明確に記載されている。

4.4.5 VPN Gate Client の実装

本研究では、4.3.3 項で述べた SoftEther VPN Client の GUI に、VPN Gate リストサーバから取得されたサーバリストを表示する画面を追加することで、VPN Gate Client を実装した (図 4.2)。このリスト表示画面には、4.3.4 項で述べた間接的サーバリスト転送プロトコルのクライアント機能が実装されている。

さらに、本研究では、VPN Gate Server の機能を VPN Gate Client ソフトウェア内に埋め込んだ。このアイデアは、BitTorrent のような P2P ファイル共有アプリケーションからヒントを得た。P2P ファイル共有アプリケーションにおいては、P2P ネットワークに接続されるクライアントは、同時にサーバとしても機能し、P2P ネットワークに貢献をする。これと同様に、VPN Gate Client は、VPN Gate Server としても機能し、VPN Gate ネットワークに貢献をする。ただし、VPN Gate Client 内の VPN Gate Server 機能は、デフォルトで無効とされている。ボランティアになりたいユーザは、この機能を手動で有効にすることができる。この機能は、VPN Gate Client が他の VPN サーバに接続中は、自動的に無効になる。

4.4.6 VPN Gate Client 配布パッケージの動的生成

4.3.4 項および 4.3.5 項で述べたように、VPN Gate ユーザは、最初に VPN Gate Client をダウンロードサーバから直接、または HTTP 中継機能を経由して間接的にダウンロードする。VPN Gate Client のダウンロードサーバは、ダウンロード要求に応じる都度、新しい ZIP ファイルのパッケージを生成する。各 ZIP パッケージの内容のうち、VPN Gate Client のバイナリファイルは固定されたバイト列である。一方、各 ZIP パッケージには、4.3.4 項で述べた初期サーバリストが含まれる。この初期サーバリストの内容は、ダウンロードに応じる度に異なるものが動的生成され、格納される。ZIP ファイルの先頭部分は、ランダムなファイル名およびランダムなデータから始まる。このようにして、VPN Gate Client の配布 ZIP パッケージがダウンロードされようとする TCP トラフィックの特徴点を減少させることにより、検閲用ファイアウォールが VPN Gate Client の配布 ZIP パッケージを検出することを困難にする。

ダウンロードサーバがダウンロード要求を受け付ける都度、一時的な ZIP ファイルを作成することは、サーバのディスク I/O や CPU 使用率を増加させることにつながる。このような負荷を減少させるため、本研究では、ダウンロードサーバに軽量かつメモリ上で動作する ZIP 生成器を実装した。各ダウンロードリクエストごとに、この ZIP 生成器はごく少量のサーバ上のメモリおよび CPU 時間しか消費しない。

4.5 評価

本研究では、2013 年 3 月 8 日に VPN Gate Web サイトを立ち上げ、VPN Gate Server および VPN Gate Client ソフトウェアの配布を開始した。この節では、VPN Gate を公開した後の 6 ヶ月間における経験をもとに、4.1 項で述べた本研究の目的の達成状況を示す。この節で述べる評価は、2013 年 3 月 8 日から 2014 年 2 月 4 日までの期間に関して行う。2014 年 2 月 5 日以降の出来事については、4.5.8 項で述べる。

4.5.1 ユーザおよびボランティアの統計情報

図 4.6 に、1 日あたりの VPN 接続数を示す。また、図 4.7 に、1 日あたりの VPN 接続元のユニーク IP アドレス数を示す。例えば、2013 年 8 月 29 日には、合計 464,000 回の VPN 接続が、88,000 個のユニークな IP アドレスから確立された。したがって、この日は、1 個のユニークな接続元 IP アドレスあたり平均 5.3 回の VPN 接続が行われたことになる。図 4.8 は、すべての VPN Gate Server を経由するトラフィックの合計帯域を示す。

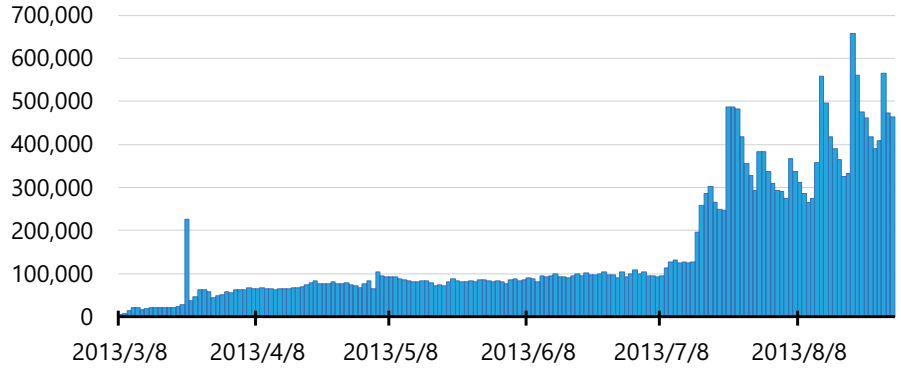


図 4.6: 1 日あたりの VPN 接続数

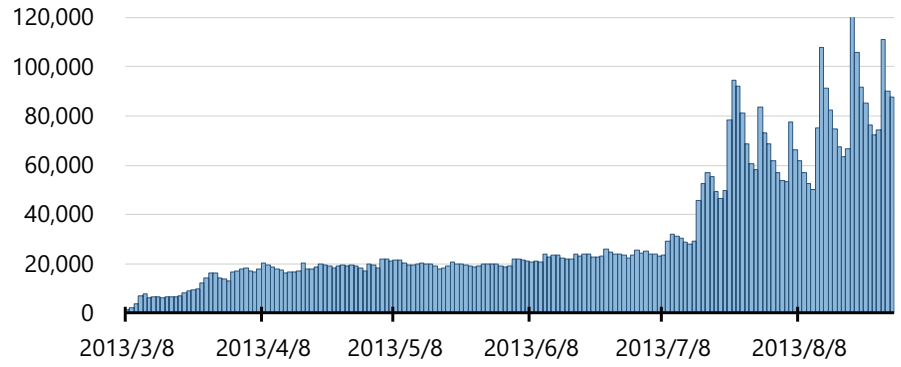


図 4.7: 1 日あたりの VPN 接続元のユニーク IP アドレス数

2013 年 8 月 29 日において、合計 1.6Gbps のトラフィックの発生を観測した。合計帯域は、ユーザ数やボランティア数が増加するに従って、安定的に増加している。

表 4.1 は、2013 年 8 月 30 日までの記録における、VPN 接続回数を元にした、接続元上位 10 ヶ国のリストである。表 4.2 は、2013 年 8 月 30 日までの記録における、VPN 経由の通信データ量を元にした、接続元上位 10 ヶ国のリストである。いずれのリストにも、中国、タイおよびイランが登場する。これらの国には、インターネット検閲用ファイアウォールが設置されている。したがって、VPN Gate はこれらの国のインターネットユーザが検閲用ファイアウォールを回避することに貢献している。表 4.2 にはまた、韓国、アメリカ合衆国、日本および台湾が含まれる。韓国においては、高速なインターネット接続回線が普及していることから、データ転送量としては第 1 位となっているが、VPN 接続回数においては第 7 位に留まっている。一方で、タイおよびイランにおいては、VPN 接続回数は極めて多いにもかかわらず、データ転送量は比較的少ない。これは、これらの国々の大半のユーザは低速なインターネット接続回線を用いているためであると考えられ

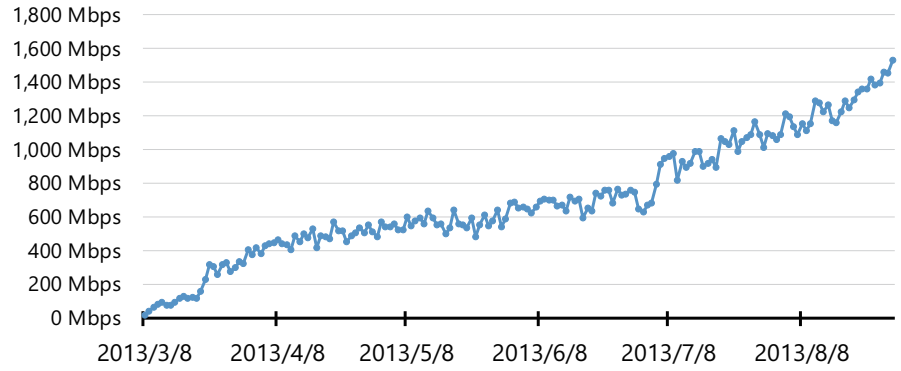


図 4.8: すべての VPN Gate Server を経由するトラフィックの合計帯域

る。これらの結果から、通信速度におけるボトルネックは主に VPN Gate のソフトウェアや VPN Gate Server 側にあるのではなく、クライアントの環境側にあることが分かる。

最初の 175 日間で、127 個の国または地域において、合計 16,523 台のボランティアサーバが立ち上げられたことを観測した。これらのサーバは、この期間中に 108,663 個のユニークな IP アドレスとしてインターネット上に現れた。

図 4.9 に、2013 年 4 月 5 日から同年 8 月 30 日までの間における、VPN サーバの IP アドレスが変化した/変化しなかった数を示す。グラフ内の青色の部分は、同一の VPN サーバが、前日から IP アドレスが変化しなかった台数を示す。一方、オレンジ色の部分は、同一の VPN サーバが前日から IP アドレスが変化した台数を示す。例えば、2013 年 8 月 30 日には、3,935 台の IP アドレスが変換しなかったサーバがあり、2,363 台の IP アドレ

表 4.1: 接続元クライアントの国・地域ごとの VPN 接続回数ランキング

順位	国・地域	VPN 接続回数	全体に占める割合
1	台湾	7,253,003 回	23 %
2	中国	3,974,954 回	15 %
3	タイ	3,841,947 回	14 %
4	イラン	2,281,446 回	8 %
5	日本	1,768,716 回	6 %
6	ベトナム	1,399,833 回	5 %
7	韓国	1,373,906 回	5 %
8	インドネシア	742,640 回	3 %
9	アメリカ合衆国	589,148 回	2 %
10	香港	466,265 回	2 %
	その他 190 の国・地域	3,536,359 回	13 %
	合計	27,228,217 回	100 %

スが変化したサーバがあった。このことは、前日から IP アドレスが変化したサーバが、全体のサーバのうち 38% を占めていることを示す。平均的に、約 40% の VPN サーバが、毎日 IP アドレスを変化させている。このように、IP アドレスが頻繁に変化することは、インターネット検閲用ファイアウォールが設置されている国からの VPN サーバへの到達率を向上させることに貢献している。

表 4.3 に、2013 年 8 月 30 日 15:00 (GMT) において稼働していた 2,800 台のボランティアサーバの地理的な分散を示す。IP アドレスから地域へのマッピングは、IP アドレスの割り当て情報を元に行った。全体のうち 77% のボランティアサーバは、韓国、日本、ベトナム、アメリカ合衆国およびロシアに分布していた。

各 VPN サーバが接続されているインターネット接続回線の品質の測定を行った。図

表 4.2: 接続元クライアントの国・地域ごとの VPN データ転送量ランキング

順位	国・地域	データ転送量	全体に占める割合
1	韓国	460.0 TB	35 %
2	中国	193.4 TB	15 %
3	アメリカ合衆国	145.7 TB	11 %
4	日本	111.1 TB	8 %
5	台湾	90.4 TB	7 %
6	イラン	45.1 TB	3 %
7	香港	28.2 TB	2 %
8	マレーシア	26.3 TB	2 %
9	ベトナム	25.8 TB	2 %
10	フランス	18.0 TB	1 %
	その他 190 の国・地域	187.1 TB	13 %
	合計	1,331.1 TB	100 %

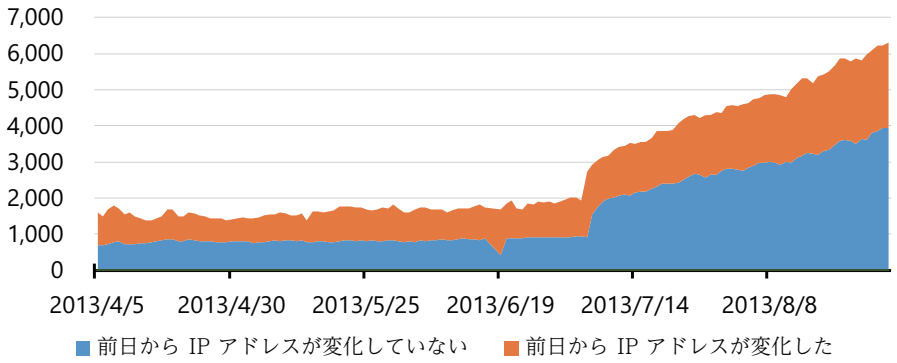


図 4.9: VPN サーバの IP アドレスが変化した/変化しなかった数

4.10 に、2013 年 8 月 30 日時点における各 VPN サーバと Google Public DNS サーバ (IP アドレス: 8.8.8.8) とのラウンドトリップタイム (RTT) の分布を示す。Google Public DNS サーバは世界中に分散配置されているので、この IP アドレスとの間の RTT の値から、各 VPN サーバが接続されているインターネット回線のラストワンマイルの遅延品質を推測することができる。多くの VPN サーバの RTT は、100 ミリ秒またはそれ未満であった。このことは、多くの VPN サーバがインターネットに対して比較的高速な回線で接続されていることを示す。図 4.11 に、各 VPN サーバと日本に設置した速度測定サーバとの間の TCP を用いた速度測定結果を示す。50% 以上の VPN サーバで、5Mbps またはそれ以上高速な結果が観測された。すべてのボランティアの回線の帯域幅の容量の合計は、70Gbps 程度であると見積もられる。図 4.8 で示したように、実際の消費帯域は 1.6Gbps であったので、ボランティアの回線の帯域幅にはまだまだ余裕がある。

表 4.4 に、2013 年 8 月 30 日時点の各 VPN サーバのインターネット接続種別の検出結果を示す。このデータから、すべての VPN サーバのうち 72.8% が NAT 装置の内側に設置されていることが分かる。これは、4.4.2 項で述べた NAT 装置との親和性を実現する

表 4.3: 2013 年 8 月 30 日時点で存在する VPN Gate Server の地理的な分散

順位	国・地域	サーバ台数	全体に占める割合
1	韓国	841 台	30 %
2	日本	637 台	23 %
3	ベトナム	444 台	16 %
4	アメリカ合衆国	181 台	6 %
5	ロシア	119 台	4 %
6	フランス	57 台	2 %
7	タイ	51 台	2 %
8	イギリス	41 台	1 %
9	インドネシア	38 台	1 %
10	カナダ	29 台	1 %
	その他 66 の国・地域	362 台	13 %
	合計	2800 台	100 %

表 4.4: 2013 年 8 月 30 日時点の各 VPN サーバのインターネット接続種別の検出結果

	サーバ台数	割合
直接接続 (NAT の内側ではない)	3,884 台	27 %
NAT の内側 (UPnP によるポート開放)	7,384 台	52 %
NAT の内側 (UDP ホールパンチングによるポート開放)	3,006 台	21 %
合計	14,274 台	100 %

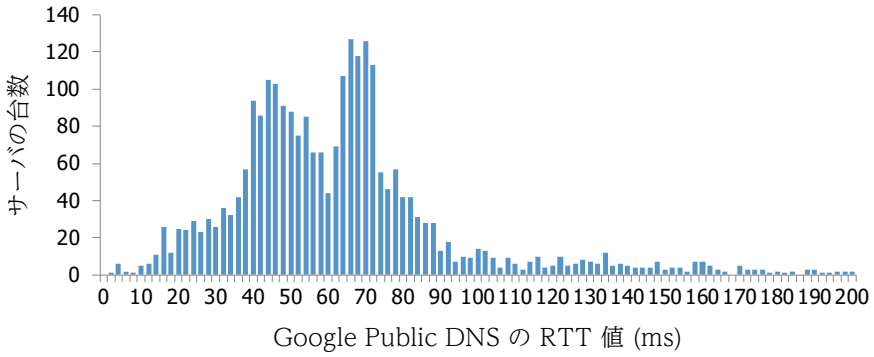


図 4.10: 2013 年 8 月 30 日時点における各 VPN サーバと Google Public DNS サーバ (IP アドレス: 8.8.8.8) とのラウンドトリップタイム (RTT) の分布

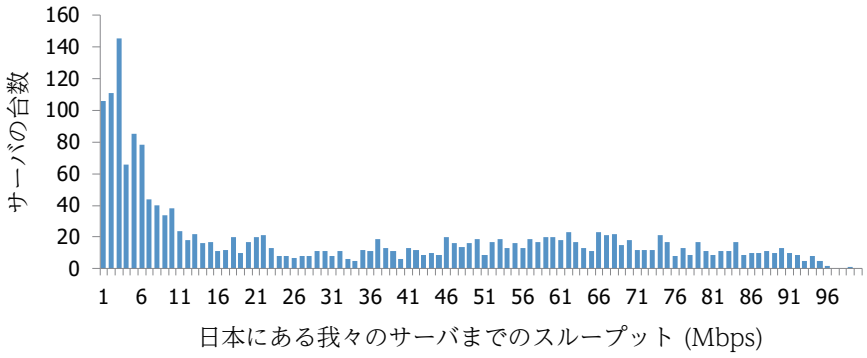


図 4.11: 各 VPN サーバと日本に設置した速度測定サーバとの間の TCP を用いた速度測定結果

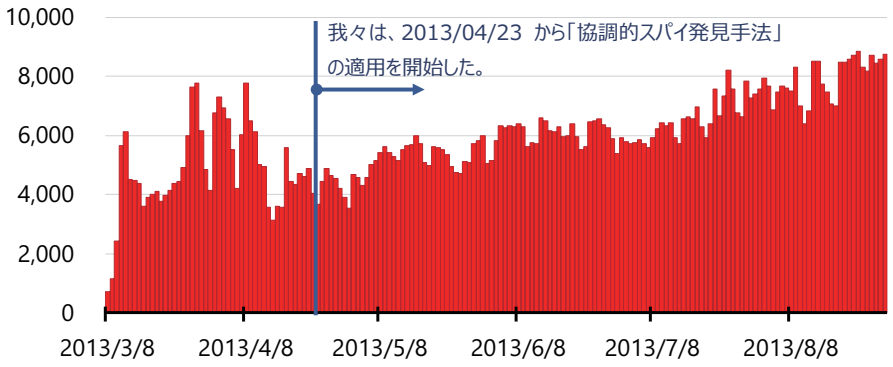


図 4.12: 中国からのユーザによる接続数 (1 日あたりユニーク IP アドレス数)

機能が、正しく動作していることを示す。

4.5.2 中国からのユーザ

中国の GFW 管理局は、2013 年 3 月 12 日から、VPN Gate リストサーバの IP アドレスを GFW の遮断リストに追加した。また、同時期に VPN Gate サーバリストに掲載されているすべての VPN サーバの IP アドレスを遮断する試みを開始した。それにも関わらず、図 4.12 に示されているように、中国からのユーザによる接続数は継続的に増加した。このグラフには、協調的スパイ発見手法によって発見されたスパイコンピュータの数は含まれていない。2013 年 8 月 29 日時点で、全世界からの VPN 接続の接続元ユニーク IP アドレス数の 10% にあたる、8,000 個の中国の IP アドレスを接続元とする VPN 接続が確立された。このことは、本研究における検閲用ファイアウォールに対する遮断耐性を実現するための手法が効果的に機能していることを裏付けている。

2013 年 3 月 22 日以降の GFW による遮断レート (VPN サーバのうち GFW によって遮断された割合) を調査した。図 4.13 のとおり、当初は GFW 管理局は VPN サーバを効果的に遮断することに成功した。この時点では、中国から到達可能な VPN サーバは 30% のみとなった。その後、無実の IP アドレス混入手法および協調的スパイ発見手法の使用を開始したところ、遮断レートは減少した。2013 年 6 月 19 日には、78.5% のサーバが中国から到達可能となった。2013 年 8 月末においては、経常的に、60% - 70% のサーバが中国から到達可能となった。したがって、検閲用ファイアウォールがある国のユーザは、サーバリスト内からランダムに 2 個のサーバを選択すると 84% 以上、ランダムに 3 個のサーバを選択すると 93% 以上の確率で、到達可能な VPN サーバに接続できるようになった。2013 年 8 月 8 日には、GFW による遮断レートが一旦急激に減少した。これは、GFW に何らかの技術的な問題が発生したことが原因であると推測される。

結果として、本研究では、VPN Gate Server において、中国の GFW に対する強い遮断耐性を実現したということができる。この遮断耐性の実現は、本研究において、無実の IP アドレスの混入および協調的スパイ発見の 2 つの手法を適用したことによるものである。また、各 VPN サーバの IP アドレスが高頻度で変化する現状も、良好な遮断耐性の実現にとって有利に作用している。

4.5.3 中国の Great Firewall 管理局との間のいたちごっこ

本研究において、VPN Gate を公開した後、無実の IP アドレスの混入手法および協調的なスパイ発見手法を適用することにより遮断レートを減少させるまでの間に、VPN

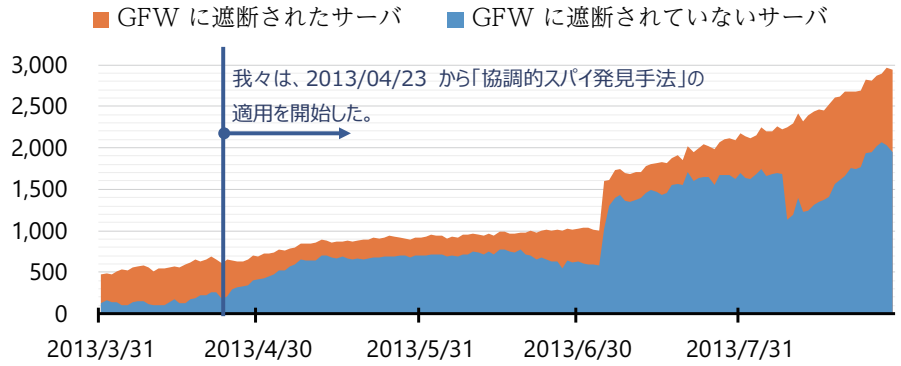


図 4.13: 中国の GFW によって遮断されている/遮断されていないサーバ数の推移

Gate と中国の Great Firewall 管理局との間では、いたちごっこが実施された。ここでは、そのいたちごっこのあらましを時系列に述べる。

4.5.3.1 2013 年 3 月 8 日: VPN Gate の公開

VPN Gate の Web サイトを立ち上げ、サーバおよびクライアントソフトウェアをリリースした。この公開日は、金曜日であった。最初の 4 日間は VPN Gate の Web サイトは GFW によって遮断されていなかったので、多くの中国からのユーザが、VPN Gate を見つけて使用し始めた。2013 年 3 月 11 日には、中国からのクライアントの接続元ユニーク IP アドレス数は 5,663 個に達した。GFW 管理局は土曜日および日曜日が休業日であり、何も対策をしなかったのではないかと推測される。

4.5.3.2 2013 年 3 月 11 日: GFW が VPN Gate リストサーバを遮断

GFW 管理局は、VPN Gate リストサーバの IP アドレスを遮断した。これ以降、中国のユーザは、VPN Gate の Web サイトにアクセスしたり、VPN Gate Client をダウンロードしたりすることができなくなった。中国の中には、4.3.5 項で述べた、HTTP 中継機能を経由してアクセス可能なミラーサイトの URL 一覧を、中国国内の SNS Web サイト (例: Weibo) を用いて拡散し始めるユーザがいた。このような HTTP 中継機能を経由してアクセス可能なミラーサイトにより、中国国内のユーザは、引き続き VPN Gate リストサーバの Web サイトにアクセスしたり、VPN Gate Client をダウンロードしたりすることができた。一度 VPN Gate Client のダウンロードに成功したユーザは、間接的サーバリスト転送プロトコルを用いて、VPN Gate Client に最新のサーバリストを継続的に取り込むことができた。

4.5.3.3　2013 年 3 月 12 日: GFW による VPN サーバの自動遮断の開始

GFW 管理局は、VPN Gate リストサーバから、稼働中の VPN サーバの一覧を取得し、リスト内にあるすべての IP アドレスを GFW の遮断リストに登録する行為を開始した。2013 年 3 月 12 日および 13 日においては、GFW 管理局は、1 日あたり 2 回、このような処理を実施した。2013 年 3 月 14 日には、1 日あたり数回この処理を実施するようになった。GFW 管理局がこの時点でこのような遮断処理を自動化するための特製ツールを実装したのではないかと推測される。この反応結果は、GFW 管理当局が常にファイアウォール回避サービスを発見しようとしていること、およびそのようなサービスが新たに開始された後、わずか 4 日で自動遮断のための特製ツールを開発し運用を開始する能力を有すること、の 2 点を明らかにした。

4.5.3.4　2013 年 3 月 13 日: GFW のスパイコンピュータの 1 個の IP アドレスを発見

本研究で用意した IP ネットワークに 32 個の IP アドレスを用意し、それぞれの IP アドレス上で、VPN Gate Server を動作させないホストを稼働させた。そして、VPN Gate リストサーバに、リストを要求してきたクライアントの IP アドレス (32 ビット) を 2 進数に変換し、これら 32 個の各 IP アドレスをサーバリストに含めて結果を返却するかどうかを、この 2 進数のビットの各位の値によって決定するようなプログラムを急いで仕込んだ。このように、複数の IP アドレスのうちいずれを返却するリストサーバに含めるかを、リスト要求元の IP アドレスを元に決定する仕組みは、一種のステガノグラフィ暗号である。例えば、リスト要求元の IP アドレスが 1.2.3.4 の場合は、32 個の IP アドレスのうち、7 番、14 番、21 番、27 番の IP アドレスをリストに含める。約 30 分後、GFW はステガノグラフィとして埋め込んだ IP アドレスのうち一部をブロックした。ブロックされた IP アドレスをもとに計算をすると、サーバリストを取得したスパイコンピュータの IP アドレスは 210.72.128.200 であると推測された。Whois データベースによると、この IP アドレスは、中国科学技術院が運営する China Science and Technology Network (CSTNET) によって運用されているネットワークにあることが分かった。

4.5.3.5　2013 年 3 月 14 日: GFW が海外のクラウドサーバを利用してサーバリストのクロールを開始

GFW 管理局であると思われる IP アドレスを遮断したところ、GFW 管理局は、Amazon EC2 および Gorilla Server を用いてサーバリストを自動取得する処理を開始した。この際、サーバリストを取得する際の HTTP リクエストにおける User agent の値は、図 4.14 に示すように、”Python-urllib” であった。GFW 管理局は、一定間隔ごとに VPN サー

ID	Access Date	Client FQDN	URL	User Agent
3312453	3/23/13 7:40 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3312674	3/23/13 7:41 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3313273	3/23/13 7:45 PM	198-136-27-242.static.gorillaservers.com	http://www.vpngate.net/	Python-urllib/1.17
3313385	3/23/13 7:45 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3313579	3/23/13 7:46 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3314469	3/23/13 7:50 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3314708	3/23/13 7:51 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3315395	3/23/13 7:55 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3315642	3/23/13 7:56 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3316250	3/23/13 8:00 PM	198-136-27-242.static.gorillaservers.com	http://www.vpngate.net/	Python-urllib/1.17
3316252	3/23/13 8:00 PM	198-136-27-242.static.gorillaservers.com	http://www.vpngate.net/cn/	Python-urllib/1.17
3316383	3/23/13 8:00 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3316570	3/23/13 8:01 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3317306	3/23/13 8:05 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3317533	3/23/13 8:07 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3318339	3/23/13 8:10 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3318553	3/23/13 8:12 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3319069	3/23/13 8:15 PM	198-136-27-242.static.gorillaservers.com	http://www.vpngate.net/	Python-urllib/1.17
3319072	3/23/13 8:15 PM	198-136-27-242.static.gorillaservers.com	http://www.vpngate.net/cn/	Python-urllib/1.17
3319236	3/23/13 8:15 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3319480	3/23/13 8:17 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3320192	3/23/13 8:20 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3320439	3/23/13 8:22 PM	ec2-50-16-163-135.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17
3321185	3/23/13 8:26 PM	ec2-23-20-4-19.compute-1.amazonaws.com	http://www.vpngate.net/en/	Python-urllib/1.17

図 4.14: 米国の Amazon EC2 および Gorilla Server で動作している中国 GFW 管理局のスパイコンピュータからのサーバリスト定期取得用のアクセス

バリストを自動取得した。これらの特徴を用いることにより、GFW 管理局のスパイコンピュータとそれ以外の一般のユーザとを区別することが可能であった。しかしながら、GFW 管理局はいつでもこのような特徴を消すことができるので、このような特徴を用いてスパイコンピュータを識別する方法は、長期的には利用できないと考えた。GFW 管理局は、VPN サーバのうち大半の IP アドレスを自動取得することに成功し、これらの IP アドレスは GFW の遮断リストに投入された。このような GFW の自動遮断プロセスの稼働が開始した後は、すべての VPN サーバのうち約 80% ものサーバが、中国から到達不能になった。

4.5.3.6 2013 年 3 月 14 日: 無実の IP アドレスの混入を開始

すぐさま、筑波大学の有する無関係の IP アドレスを複数個、無実の IP アドレスとして VPN サーバリストに混入する実験を開始した。その結果、無実の IP アドレスを混入してから約 30 分後に、その IP アドレスが中国国内から到達不能になることが観測された。この実験を 4 時間余りに渡って繰り返した。毎回、新たに混入した無実の IP アドレスは 30 分またはそれ以下の時間が経過すると、中国国内から到達不能となった。このことは、GFW 管理局はこの時点で VPN Gate のサーバリストを信頼しており、サーバリスト内にある IP アドレスを GFW の遮断リストに投入する前には、その IP アドレスが真に VPN Gate Server の IP アドレスであるか否かの検証を行っていないことを意味する。言い換えれば、著者はこの時点では GFW の遮断リストを自由にコントロールする力

を有していたということができる。

4.5.3.7　2013 年 3 月 16 日: GFW 管理局がクローラを用いた VPN Gate の自動遮断を一旦中断

著者が VPN リストサーバに色々な無実の IP アドレスを混入させる実験をしていたところ、GFW 管理局は最終的に、返却されるサーバリストは信頼できず、時々無関係な IP アドレスが混入されている可能性があることに気付いたようである。GFW 管理局は、クローラを用いた VPN Gate の自動遮断を中断した。その結果、中国国内から、一旦すべての VPN Gate のボランティアサーバにアクセスできるようになった。このときから 4 日間、GFW は VPN Gate のボランティアサーバを一切遮断しなくなった。

4.5.3.8　2013 年 3 月 20 日: GFW 管理局が IP アドレスの検証を開始

GFW 管理局は、GFW の遮断リストに IP アドレスを追加する前に、その IP アドレス上で真に VPN Gate Server が稼働しているかどうかの検証を開始するようになった。

4.5.3.9　2013 年 4 月 24 日: 協調的スパイ発見手法の適用を開始

VPN Gate において、4.3.3 項で述べた協調的スパイ発見手法の適用を開始した。

4.5.3.10　2013 年 9 月 2 日: 遮断の中止

2013 年 9 月 2 日に、GFW による VPN Gate の遮断機能は不安定になった。まず、GFW は突然すべての VPN Gate のサーバに対する遮断をやめた。数時間後、GFW は再度 VPN Gate に対する遮断を開始した。その後、遮断されたり、遮断がされなくなったりすることが数日間の間に何度か繰り返された。

2013 年 9 月 5 日には、GFW はすべての VPN Gate サーバに対する遮断を完全にやめた。しかしながら、調査用パケットの送付は、その後も継続的に続いている。GFW が遮断を停止した理由は、不明である。

その後も、VPN Gate のユーザ数とボランティア数は継続的に増加している。2014 年 2 月 4 日には、1 日あたり 5,200 台のボランティアサーバが、1 日あたり 1,049,000 回の VPN 接続 (156,000 個のユニークな IP アドレス) を処理した。これには、中国からの 1 日あたり 123,000 回の VPN 接続 (16,000 個のユニークな IP アドレス) が含まれる。2016 年 7 月時点で、1 日あたり中国の 34,000 個のユニーク IP アドレス、および全世界からの 234,000 個のユニーク IP アドレスから利用されており、全世界で平均 6.7 Gbps のトラフィックを分散処理している。

4.5.4　スケーラビリティ

VPN Gate は、通信の中継をボランティアサーバに依存している。ボランティアの数が増加するに従い、合計の帯域幅も増加する。これらのボランティアサーバのサーバリストは、VPN Gate リストサーバに集約される。また、ユーザは VPN Gate リストサーバからサーバリストを取得することになる。したがって、VPN Gate のスケーラビリティは、VPN Gate リストサーバに依存することになる。

現在のところ、VPN Gate リストサーバのインスタンスは、合計で 3 台の Web サーバ、1 台のデータベースサーバ、1 台のステータス監視サーバおよび 1 台のログ分析サーバによって構成されている。これらのサーバは、筑波大学のキャンパスネットワークを経由して、インターネットに接続されている。これらのサーバのうち、Web サーバのみがボランティアのサーバおよびユーザからのリクエストを処理する。これらの Web サーバ上では、同一のアプリケーションが動作しているため、Web サーバの台数を増加させることにより、全体のパフォーマンスを容易にスケールアウトすることができる。

データベースサーバは、Intel Xeon E3-1230 3.2-GHz プロセッサ (富士通 PRIMERGY TX100 S3) 上で Microsoft SQL Server を稼働させている。このサーバには 1.0MB の L2 キャッシュ、32GB のメインメモリおよび 2 台の SSD ドライブが搭載されている。このデータベースサーバの負荷は、Web サーバの負荷よりも軽い。データベースサーバの CPU 使用率は平均約 5% であり、ディスク I/O の帯域幅は平均約 1.3MB/s である。現在のデータベースサーバのハードウェアのみで、現状の 10 倍の負荷を処理することができると推測される。

データベースサーバが過負荷となった場合は、ボランティアサーバを複数のグループに分割し、それぞれのグループごとにデータベースサーバを用意することで対応できる。これらの分割されたグループごとに、協調的スパイ発見手法におけるログの分析処理を実行することができる。このようなグループ分割は、同時に、可用性も向上させる。

3,000 台の VPN サーバに対するステータス監視処理は、10 分間以内で完了する。VPN サーバの台数が増加した場合は、ステータス監視のためのサーバを複数台設置し、タスクを分割することで対応できる。

4.5.5　Tor との比較

2013 年 8 月 29 日時点で、VPN サーバ数は約 3,000 であり、これは当時の Tor ノードの数に匹敵する。当時の Tor ネットワークは、4,000 のリスト化されたサーバおよび 2,000

の隠されたブリッジノードから構成されていた。

中国からの VPN Gate のユーザ数は、中国からの Tor のユーザ数よりも多い。2013 年 8 月末時点において、中国からは 1 日あたり 9,000 個のユニークな IP アドレスから VPN Gate への接続があった。Tor メトリック・サイト [*3] によれば、中国からの Tor のユーザ数は 3,000 程度であるとされる。このように、中国からの多くのユーザを得ることを実現できた要因は、無実の IP アドレスの混入手法および協調的スパイ発見手法を用いることで中国の検閲用ファイアウォールに対する遮断耐性を高めたことにある。Tor には、ノード間においてこのような協調動作を実施し、検閲用ファイアウォールに対する遮断耐性を高める機能がない。

VPN Gate は、Tor に対する別の優位性もある。VPN Gate は IP パケットを伝送することができる VPN トンネリング機能を提供する。ユーザはアプリケーションのプロキシサーバの設定を行うことなく、また、プロキシサーバに対応していないアプリケーションであっても、VPN Gate を用いて通信することができる。

4.5.6 ボランティアの地域の法令や LAN のポリシーへの違反リスクの軽減と負荷の削減手法

従来の VPN サーバは、インストールや設定が複雑なため、VPN 中継サーバを立ち上げようとするユーザは、実現したい中継サービスの内容相応の努力を必要とした。一方、本研究においては、4.4.1 項で述べたように、ボランティアは VPN Gate Server を、従来よりも簡単な操作で自分のコンピュータにインストールすることができる。また、4.4.5 項で述べたように、ボランティアは簡単な操作で VPN Gate Client 内に組み込まれている VPN Gate Server 機能を有効にすることができる。簡単に VPN の中継サーバを立ち上げることができることは、ボランティアの数を増やすことにつながる反面、ボランティアは、ローカルルールによっては、以下のような場合にペナルティを受ける可能性がある。

ボランティアの居住する国の法令上の許可が必要な可能性

日本の法令では、営利を目的としない VPN 中継サーバを立ち上げる際には、政府への届出や、政府からの許可の取得は不要である[*4]。また、VPN Gate の暗号通信や検閲回避などに関して規制する法令も存在しない。しかし、国によっては、VPN 中継

[*3] https://metrics.torproject.org/graphs.html

[*4] 日本の法令では、営利を目的とする VPN 中継サーバを立ち上げる際には、電気通信事業法に基づき、総務大臣への届出が必要である。本研究では、2013 年 3 月に総務省に VPN Gate の趣旨を説明し、ボランティアが電気通信事業法上の届出を要するかどうかの見解を求めた。その結果、VPN Gate のボランティアとなることは営利を目的としていないと考えられることから電気通信事業法上の届出は不要であるとの回答を得た。

サーバを立ち上げる際に、何らかの許可が必要とされている可能性がある。そこで、ボランティアの責任で、居住国の法令の遵守をするよう注意してもらう必要がある。法令遵守が不十分である場合は、ボランティアがペナルティを受ける可能性がある。

VPN サーバが設置される国の法令上の許可が必要な可能性

ボランティアが VPN Gate Server を立ち上げる操作を行う国と、VPN Gate Server が実行されるコンピュータが設置されている国が異なる場合がある。たとえば、海外のクラウドサーバ上に VPN Gate Server を立ち上げるケースが存在する。この場合、ボランティアは、VPN Gate Server が実行されるコンピュータが設置されている国の法令を遵守し、必要な許可を受けなければならない。法令遵守が不十分である場合は、ボランティアがペナルティを受ける可能性がある。

VPN サーバを社内 LAN に設置する場合のポリシーへの抵触の可能性

一部の組織の社内 LAN などでは、社内 LAN 上に、任意の第三者の通信を中継するためのコンピュータを設置することを禁止している可能性がある。たとえば、社内 LAN に誰でもアクセスできる無線 LAN アクセスポイントを設置することを禁止している場合がある。このような環境で社内 LAN に VPN Gate Server を立ち上げるためには、ボランティアは、社内 LAN の管理者に事前に許可を得なければならず、無許可の場合、ボランティアがペナルティを受ける可能性がある。

回線帯域を第三者のために提供することを禁止している ISP との契約違反の可能性

一部の ISP と、一部の ISP のユーザの間のサービス利用契約において、回線帯域を第三者のために提供することを禁止している場合がある。この場合、VPN Gate Server をその ISP の回線で立ち上げたボランティアは、ISP との間の契約に違反することになり、追加料金の請求または契約解除などのペナルティを受ける可能性がある。

ボランティアが上記のようなリスクを避けるためには、ボランティアの地域の法令や LAN のポリシーへの準拠の励行をすることが望ましい。そのためには、何段階かの警告と確認を表示し、すべて明示的に確認がなされた場合にのみ、VPN 中継機能が動作することが望ましい。ローカルの法令やポリシーへの遵守の励行については、VPN Gate Server のインストール時に「重要事項説明」というメッセージを表示するようにした。メッセージは、日本語、英語、中国語の 3 ヶ国語で用意した。しかし、このメッセージを読まないボランティアがいることが予想された。そのため、VPN Gate Server のプログラムを工夫し、インストールしただけでは VPN 中継機能が動作しないようにした。VPN 中継機能を起動するためには、さらに、VPN Gate Server の GUI で表示される図 4.15 の画面

図 4.15: VPN Gate Server 機能の動作を開始させるための画面 (日本語、英語、中国語)

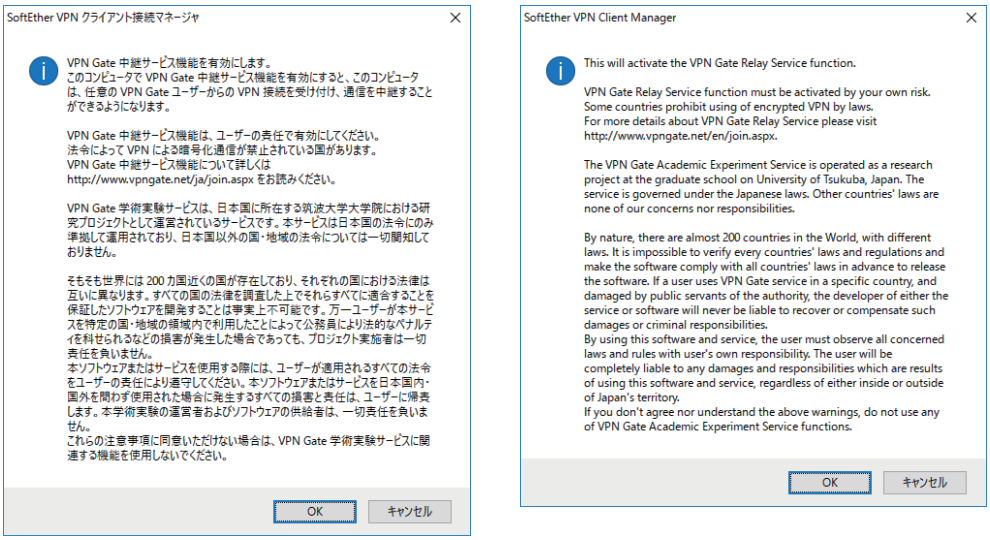


図 4.16: ユーザの責任で地域の法令を遵守するよう警告するメッセージ (日本語・英語)

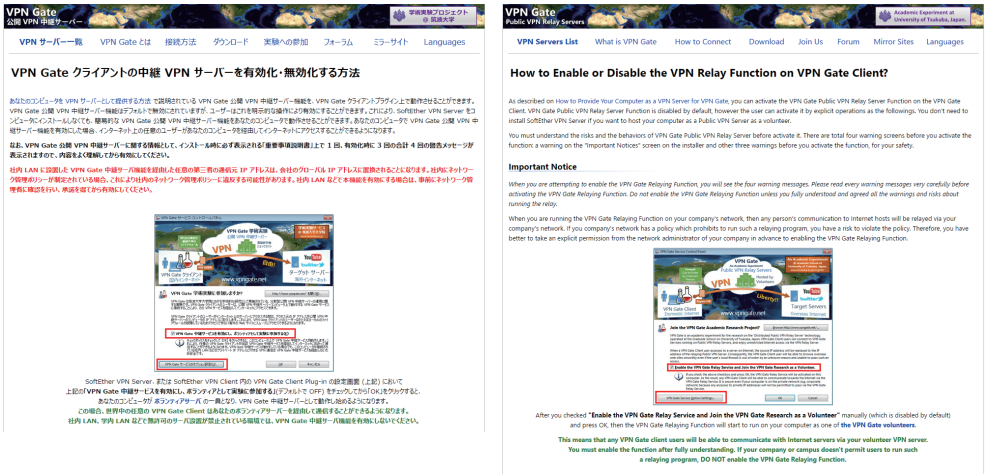


図 4.17: VPN Gate Server 機能に関する注意事項を掲示した Web サイト (日本語・英語)

の説明書きを読み、チェックボックスを有効にする必要があるようにした。この注意書きは、日本語、英語、中国語の 3 ヶ国語で用意した。さらに確認手順を追加するため、図 4.16 の警告メッセージを表示した。この警告メッセージに示されている Web サイト上に、図 4.17 のような、目立つ注意書きを掲載した。VPN Gate Server の GUI では、これらの複数段階の警告メッセージすべてに対して明示的な同意があった場合のみ、VPN 中継機能が動作するようにした。

しかしながら、これらに同意して立ち上げられたボランティアのサーバを経由した通信に関する外部からの照会は、その日時にその IP アドレスの割り当てを受けていたボランティアに寄せられることとなるため、ボランティアにとって、照会への対応が負荷となるという問題がある。この問題の解決策として、ボランティアのサーバを、2 種類に分ける方法が考えられる。VPN 接続を受け付ける中継専用サーバと、それらのサーバに接続した VPN クライアントがインターネット上のホストにアクセスする際に NAT として使用される出口専用サーバとに分ける方法である。ボランティアは、中継専用サーバと、出口専用サーバのどちらをホストするかを選択できる。従来どおり、両方の機能を有するホストを立ち上げることも可能である。この場合、クライアントの通信は、まず中継専用サーバを経由して、出口専用サーバに届き、次に、出口専用サーバの IP アドレスを用いてインターネットに送信される。外部からの照会は、出口専用サーバの IP アドレスを有していたボランティアに寄せられる。この方法により、外部からの照会をできるだけ受けたくないと考えるボランティアは、気軽に中継専用サーバのみを立ち上げることができる。一方、外部からの照会を受けても良いと考えるボランティアは、出口専用サーバを立ち上げることができる。しかしながら、この方法には、VPN 通信に関するログが、中継専用サーバと出口専用サーバとの間で分かれて保存されることになるという問題がある。出口専用サーバのログには、通信が経由した中継専用サーバの IP アドレスが記録されるが、中継専用サーバに接続をしていたクライアントの IP アドレスは記録されない。クライアントの IP アドレスは、中継専用サーバのログにのみ記録される。これにより、4.4.4 項で述べた違法行為の抑制が低下してしまう。また、この方法では、クライアントの通信は 2 台のサーバを経由するため、通信速度が低下してしまう。このように、ボランティアが照会に対応する際の負荷の軽減と、違法行為抑制や通信速度の低下の防止は、トレードオフの関係にある。今後は、ボランティアの負荷の軽減と違法行為抑制の両立を実現するため、さらに優れた解決策を考えたい。

4.5.7　課題および検討事項

違法行為を目的とするユーザは、VPN Gate を悪用して IP アドレスを隠蔽することができる。このような悪用行為は、4.4.4 項で説明したログ保存により抑制される。別の問題として、VPN サーバを稼働させているボランティアは、VPN ユーザが送受信するパケットがカプセル化解除されるところをタップしたり、パケットを改ざんしたりすることができる。この問題は、以前から存在するものである。公開プロキシや Tor ノードにおいても同じ問題が存在し、現在のところ、本研究では、有効な解決策を有していない。VPN サーバがボランティアのインターネット接続回線の帯域幅を消費するという問題もある。解決策としては、VPN サーバに帯域制御機能を組み込み、ユーザが簡単な操作で帯域を制限できるようにする手法が考えられる。

無実の IP アドレスの混入手法により混入された IP アドレスが、インターネット検閲がある国の中からアクセスできなくなるという副作用がある。これは、その IP アドレスの所有者およびその IP アドレスにアクセスしたいと考えるインターネット検閲がある国内のユーザに迷惑をかける。しかし、本研究では、この件についてこれまでにいかなる苦情も受けていない。

検閲用ファイアウォールを管理する検閲当局は、無実の IP アドレスの混入に対抗するため、ホワイトリストを作成し、そのホワイトリストに登録されている IP アドレスは検閲用ファイアウォールにおける遮断リストに投入しないことにするかも知れない。しかしながら、そのようなホワイトリストのメンテナンスは、極めて困難である。検閲用ファイアウォールが存在する国であっても、安定したインターネットアクセスは、市民およびその国を訪問する外国人の両方にとって、極めて重要である。たとえば、ある日に Yahoo! US メールの IP アドレスを混入し、翌日には Amazon EC2 の IP アドレスを混入することができる。さらに、多くの Web サーバは、Akamai [79] やその他の CDN サーバのホストする同一の IP アドレスを有している。このようなインターネットの現状を鑑みると、検閲当局が、インターネット上で遮断されてはならないホストの一覧を作成し、これを予めホワイトリストに登録しておくことは事実上不可能であると考えられる。

検閲当局が、VPN Gate ネットワークを混乱させるために、偽の VPN Gate サーバを立ち上げることが考えられる。このような偽のサーバを立ち上げる攻撃手法は、Sybil Attack と呼ばれる [19]。これらの偽の VPN Gate サーバは、偽の VPN 接続ログを、偽の接続元 IP アドレス (国内で VPN Gate を利用しているユーザの IP アドレスなどを含めるのが効果的である) などの情報とともに、VPN Gate リストサーバに伝送する。これ

により、本研究における協調的スパイ発見手法に誤作動を生じさせることが可能である。ただし、少ない台数の偽の VPN Gate サーバだけでは、ネットワーク全体を混乱させることはできない。なぜならば、本研究ではこのような少数報告を無視することができるためである。しかしながら、もし検閲当局が多数の偽サーバを立ち上げた場合は、ネットワーク全体を混乱させることができる。このような Sybil Attack の実行のためにはコストがかかり、また、目立つことになるため、検閲当局が、このような積極的な攻撃を仕掛けてくる可能性は高くないと想定される。ボットネットを利用すれば、低いコストで Sybil Attack を実行することも可能である。ボットネットにおける DDoS 攻撃によるトラフィックが、正規のトラフィックと判別することが難しいように、ボットネットによる偽の VPN Gate サーバが多数出現した場合は、正規の VPN Gate サーバと区別することは難しい。

GFW 管理局のように豊富な予算を有する検閲当局は、調査用パケットの送信元の IP アドレスとして利用可能な大量の IP アドレスプールを有している可能性がある。しかしながら、そのような IP アドレスプールは、長期的にアサインされた固定 IP アドレスブロックであると考えられる。表 4.5 に、2013 年の一部期間中に、GFW 管理局によって送付されたと考えられる調査用 IP パケットの送信元 IP アドレスの検出された個数を示す。このデータによると、GFW 管理局は、毎月、前月に使用されていた IP アドレスブロックと同一の IP アドレスブロックを、調査用 IP パケットの送信元 IP アドレスとして利用している。この結果は、GFW 管理局がわずか 4,000 個程度の固定されたクラス C の IP アドレスブロックしか有していないことを示している。

表 4.5: GFW 管理局が毎月使用した調査用パケット発信元 IP アドレスのブロック数 (/24) の推移

2013 年	IP アドレスブロック総数	新登場ブロック数	再利用ブロック数	再利用割合
3 月	2,792	2,792	0	0 %
4 月	2,645	441	2,204	83 %
5 月	1,199	103	1,096	91 %
6 月	1,509	93	1,416	94 %
7 月	1,856	235	1,621	87 %
8 月	1,792	98	1,694	95 %
9 月	1,516	92	1,424	94 %
10 月	1,168	129	1,039	89 %

4.5.8　GFW の 2014 年 2 月以降の動き

2013 年 9 月 5 日に GFW がすべての VPN Gate サーバに対する遮断を完全にやめた後、9 ヶ月後の 2014 年 6 月 8 日まで、VPN Gate サーバに対する遮断は観測されなかった。本研究では、2014 年 4 月 3 日に、国際会議 NSDI (USENIX Symposium on Networked Systems Design and Implementation) において、本研究における無実の IP アドレス混入手法および協調的スパイ発見手法に関して発表を行った。2014 年 6 月 9 日に、GFW による、VPN Gate サーバに対する新たな遮断手法が観測されるようになった。この新たな遮断手法は、GFW の機能がアップグレードされたことを示す。

そこで、本研究では、この新たな遮断手法に対する対策手法を実装し、2015 年 1 月 31 日に動作を開始させた。これらのこのことから、中国政府は、GFW に対して、検閲回避技術に対して追従し、対策技術を実装する活動を継続的に行っていることが明らかとなった。なお、GFW による新たな遮断手法および新たに実装した対策手法の内容は、本論文の範囲外である。

4.6　本章のまとめ

4.6.1　VPN Gate における検閲耐性を向上する手法について

本研究では、中国の Great Firewall (GFW) のような検閲用ファイアウォールに対する強力な遮断耐性を有する VPN 中継システムである VPN Gate の設計および実装を行った。VPN Gate においては、無実の IP アドレスの混入手法および協調的スパイ発見手法という 2 つの重要な手法を用いた。その結果、2013 年 5 月 2 日から 9 月 4 日までの間、全体の約 60% - 70% の VPN サーバが、GFW による遮断を避けることができた。検閲用ファイアウォールが存在する国のインターネットユーザは、1 台でも接続可能な VPN サーバに到達すれば、インターネットに検閲を回避して接続できるようになる。検閲当局は、すべての VPN サーバを遮断しなければ、そのようなユーザの通信の遮断を達成できない。これは、検閲当局にとって非常に難しい仕事である。

VPN Gate は、多数のボランティアに依存することにより、効率良く動作する。ボランティアのおかげで、中継機能の提供に関するコストの支出をする必要がない。代わりに、分散配置されたボランティアが、それぞれの PC において、少量の電力および帯域幅を提供することにより、VPN Gate ネットワークのために貢献している。これと比較して、検閲当局は、複雑な調査用パケットの送付プログラムなどを実装し、これらを継続的に動作

させる必要があり、このような検閲用インフラストラクチャの構築と運用には、多額のコストを支出しなければならない。

4.6.2　統合的な VPN サーバソフトウェアの有用性について

この VPN Gate に関する研究結果により、第 3 章で述べた統合的 VPN サーバソフトウェアには、ボランティアベースの大規模な分散検閲回避システムを構築するための有用性があることが明らかとなった。

VPN Gate の中継 VPN サーバソフトウェアである VPN Gate Server をできるだけ多くのボランティアに動作させてもらうためには、単一インスタンスによる複数プロトコルへの対応、インストールの容易性、システム管理者権限への非依存、通信ログの統一性および集約性、セキュリティポリシーの設定の一元化などを実現することが望ましい。

本研究では、これらの要件の実現のために、統合的な VPN サーバソフトウェアを効果的に使用した。従来の VPN サーバソフトウェアは、OS の有するネットワーク機能への依存性、システム管理者特権の必要性、動作時における OS の挙動の変更による他プロセスへの影響の発生などの問題があった。また、できるだけ多くの被検閲国のユーザに利用してもらうためには、複数の VPN プロトコルのサポートが望まれるが、従来の VPN サーバソフトウェアを使用する方式では、複数の VPN サーバを同時に稼働させる必要があった。コンピュータの操作に関して、平均的知識を有するボランティアに対して、複数の VPN サーバソフトウェアをインストールしてもらい、かつ、OS の挙動の変更による他のアプリケーションへの影響が発生する可能性を受忍してもらうことは、困難であった。本研究では、これらの問題を、検閲回避システムとしてボランティアに配布するソフトウェアに、統合的な VPN サーバソフトウェアを組み込むことにより解決することが可能であると考えた。

そこで、本研究においては、これらの要件を、SoftEther VPN Server を拡張することで実装した。統合的な VPN サーバソフトウェアである SoftEther VPN Server がサポートしている 7 種類の VPN プロトコルのうち、VPN Gate のクライアント OS として想定する Windows、Mac OS X、iOS および Android からリモートアクセスのために使用される可能性の高いプロトコルは、L2TP/IPsec、OpenVPN L3、MS-SSTP および SoftEther VPN プロトコルであった。本研究では、これらの 4 種類の VPN プロトコルを VPN Gate Server で統一的にサポートした。VPN Gate Server を立ち上げると、1 つの仮想 HUB が作成され、セキュリティ設定やユーザ認証などの設定は、これらの各 VPN プロトコルに、統一的に適用されるようにした。また、VPN Gate Server が出力するログファイルは、複

数の VPN プロトコルが使用されている場合でも、1 つのログに統一され、フォーマットも統合されるようにした。VPN 接続ログは、ファイルに保存されるほか、VPN Gate 固有のプログラムにもデータが渡るようにした。この VPN 接続ログは、協調的スパイ発見手法で利用するために、VPN Gate リストサーバに対して定期的に送付されるようにした。VPN Gate Server は、VPN 接続している複数のユーザの間で、インターネット接続を共有する機能を必要とする。1 個のインターネット接続のための IP アドレス (グローバル IP アドレスまたは NAT の内側のプライベート IP アドレス) を、複数の VPN クライアントに割り当てたプライベート IP アドレスで共有する必要がある。このために、3.3.5 項で述べたユーザモード NAT を使用した。ボランティアは、OS のネットワーク設定を変更したり、NAT を実現する他のプログラムをインストールしたりすることなく、また、OS 上での特権を使わずに VPN Gate Server を動作させることができるようになった。

また、VPN Gate のクライアントソフトウェアである VPN Gate Client にも、VPN Gate Server と同一の機能を組み込み、ユーザが VPN Gate Client を使用していないときには、VPN Gate Client が VPN Gate Server としても動作可能にした。この手法により、VPN Gate は、ボランティア数を増やすことができた。これらの結果により、VPN に関して専門知識を有さない多くのボランティアが、VPN Gate Server をインストールして VPN 中継サーバを立ち上げることができたことを示すことができた。

このように、多数のボランティアによって動作させてもらえる VPN Gate の VPN サーバソフトウェアを実装し、その機能により VPN Gate における検閲用ファイアウォールに対する強力な遮断耐性を実現することができた結果により、第 3 章で実装した統合的な VPN サーバソフトウェアの有用性を示すことができた。

4.6.3 今後の課題

インターネット検閲において、検閲能力を強化しようとする試みと、検閲回避機能を強化しようとする試みとは、本質的には、いたちごっこの関係にある。しかしながら、これは完全に平等なゲームではなく、検閲回避機能を強化しようとする試みのほうが、検閲能力を強化しようとする試みよりも有利である。VPN Gate をリリースした後、GFW 管理局との間でこのようなゲームを行うこととなり、その結果として、一時的にはこのゲームに勝利をすることができた。今後も、本研究では、GFW のような検閲用ファイアウォールについて研究を行い、また新たな検閲耐性の向上手法について研究を行うことにより、VPN Gate の遮断耐性を強化していきたいと考えている。

また、今後の課題の 1 つとしては、VPN Gate リストサーバのスケーラビリティを向上

させることが挙げられる。VPN Gate において IPv6 をサポートすることも目標の 1 つである。

本研究における、外国政府の運用する検閲用ファイアウォールに対するいかなる調査、研究および検閲回避システムの実装も、日本国内において、日本国の法令に準拠して実施した。

第 5 章

結論

第 5 章では、本研究の結論を述べ、今後の研究の発展を挙げる。

5.1 まとめ

本論文では、統合的な VPN サーバソフトウェアの実現手法、およびその応用に関する研究について述べた。

5.1.1 統合的な VPN サーバソフトウェア

統合的な VPN サーバソフトウェアとは、システム管理者が、多様な VPN プロトコルの違いを意識することなく、これらの VPN プロトコルを統一的に管理およびサポートでき、また、他の周辺プログラムにできるだけ依存せずに、1 つのプロセスとして動作する VPN サーバソフトウェアである。

従来、システム管理者は、サポートしたい異なる複数の VPN プロトコルごとに、単機能の VPN サーバソフトウェアをインストール、設定および運用する必要があった。従来の VPN サーバソフトウェアは、周辺プログラムへの依存の度合いが高く、これらの依存先の複数のプログラムを組み合わせて正しく設定する必要があった。特定の OS への依存も強かった。そのため、単一の OS 上で複数の VPN サーバソフトウェアが共存できない場合は、複数のコンピュータを用意する必要があった。複数のコンピュータを用意する代わりに、仮想マシン (Virtual Machine: VM) を利用することもできたが、VM や仮想スイッチに関する知識やトラブルシューティングの実施が必要であった。

統合的な VPN サーバソフトウェアがあれば、システム管理者は、異なる複数の VPN プロトコルをサポートする VPN サーバを、1 台のコンピュータ上で、1 個のプロセスを起動するだけで構築できる。また、複数の VPN プロトコルに対するユーザ認証、セキュリティ設定、ログの管理、IP アドレスの管理などを一元化することができる。さらに、特定

の OS に依存する部分を少なくすることにより、標準的な機能を有する幅広い OS 上で動作する。これらの特徴により、システム管理者は、VPN サーバソフトウェアのために複数の OS やコンピュータや VPN ルータアプライアンスなどを用意する必要がなくなる。

従来の VPN サーバソフトウェアは、特定の OS のカーネル内コードに依存したり、カーネルモジュールをインストールしたりする必要がある。さらに、動作には特権が必要であり、OS のネットワーク機能の挙動を書き換えるため他のプロセスに影響を生じさせる場合があった。統合的な VPN サーバソフトウェアを、ユーザモードで完結して動作するように実装すれば、実行時に、OS のネットワーク機能の挙動や設定を書き換える必要がなく、他の実行中のプロセスの通信に大きな影響を与えず動作させることができるようになる。また、一般ユーザ権限でも動作可能である。多くの OS が共通的に提供するシステムコールおよび OpenSSL などの普及している暗号化ライブラリのみに依存するように実装すれば、VPN サーバをインストールすることが容易になる。また、検閲回避システムなどのアプリケーション内に VPN サーバを組み込むことも容易になる。

5.1.2 従来手法

第 2 章で述べたように、本研究に関係する従来手法としては、以下のようなものがある。

OpenVPN、poptop および openl2tpd などの、従来の VPN サーバソフトウェアは、1 つの VPN プロトコルにしか対応していない。OpenWRT や DD-WRT、VyOS などの、複数の VPN プロトコルに対応したルータのファームウェアがある。これらは、内部で OpenVPN、poptop および openl2tpd などを起動して VPN 処理を行っている。VPN トンネル間のメッセージ交換は、Linux カーネル内のルーティングプログラムで行っている。

統合的なシステムソフトウェアとして、DeleGate がある。DeleGate は、複数のプロトコルをサポートするプロキシであり、異なるプロトコル間の変換も可能である。また、ユーザレベル OS のためのユーザレベルネットワーク機能を実現する研究がある。この研究では、通常は OS のカーネル内に実装されている TCP/IP スタックと同等のモジュールを、ユーザレベル OS、または VM プログラム内に統合している。これにより、ホスト OS 上での特権なしに、ゲスト OS が物理ネットワーク上の外部ホストと通信することを可能とする。

統合的な開発環境として、Visual Studio や Eclipse がある。これらは、複数のプログラミング言語に対する統合的な開発環境を実現する。また、プラグインを追加することで、新たな対象プログラミング言語を追加することができる。

本研究で実装した VPN サーバソフトウェアは、OpenVPN などの従来の VPN サーバ

ソフトウェアと異なり、1 つの VPN サーバソフトウェアで、複数の VPN プロトコルをサポートする。また、OpenWRT や DD-WRT、VyOS などのルータのファームウェアや、Visual Studio、Eclipse などの統合的な開発環境と異なり、複数のモジュールを静的リンク方式で結合し、1 つのプロセスにする。これにより、1 つのソフトウェアをインストールするだけで複数の VPN プロトコルを利用できる VPN サーバを構築できる。

検閲回避システムとして、The Onion Router (Tor) がある。Tor は、P2P 技術を用いて実装された、インターネットアクセスを匿名化するオーバーレイネットワークである。匿名性を重視しているため、通信は低速である。さらに、検閲ファイアウォールの管理者によるスキャンに対する耐性を有していない。

P2P システムとして、BitTorrent トラッカがある。BitTorrent のトラッカとは、BitTorrent のノード間を接続するための情報を提供する中央サーバである。本研究における VPN Gate は、同様に中央サーバを用いてボランティアの VPN サーバの一覧を提供する。

多くの研究者が、検閲耐性を実現するためのシステムに関する研究を行ってきた。Web MIXes は、匿名性の実現および検閲の防止を目的とした Web アクセス中継システムを実現するための研究である。ランダムな長さのダミーメッセージを挿入したり、大きなメッセージを Adaptive chop-and-slice アルゴリズムによって分割するなどして、検閲者による通信内容の識別を難しくする。中国の Great Firewall (GFW) を回避するための研究もある。GFW による TCP コネクションの切断手法を分析し、エンドポイント側で偽の RST パケットを無視するなどの手法で検閲を回避する。Infranet は、HTTP による Web アクセスの検閲と監視を回避するための研究である。また、キースペース・ホッピングという手法がある。これは、検閲回避のために、多数のプロキシを設置し、単一のクライアントに対しては、一部のプロキシの IP アドレスのみを回答する仕組みの検閲回避システムで使用されている。SafeWeb 匿名化システムという研究がある。SafeWeb では、Web ブラウザ上で動作する JavaScript で中継トンネル用のクライアントが実装されている。Proximax は、プロキシサーバベースの検閲回避システムであり、プロキシプールから、ユーザに適合したプロキシを選択する仕組みを提案している。Telex は、プロキシサーバベースの検閲回避システムであり、クライアントが Transport Layer Security (TLS) のヘッダに、楕円曲線暗号を用いた電子透かしを埋め込む手法である。無料や有料の VPN サービスは、検閲用ファイアウォールを回避するために広く使われている。検閲当局によって IP アドレスが遮断された場合は、IP アドレスを変更し、新たな IP アドレスをユーザにメール等で周知する必要があり、再度遮断される可能性もある。

本研究で実装した VPN Gate は、Tor と異なり、匿名性の実現を目的とせず、検閲回避

を目的としている。そのため、経由するホスト数は 1 台のみであり、複数台を経由する際のボトルネックがない。また、検閲ファイアウォールの管理者によるスキャンに対する耐性を有している。Tor、SafeWeb および Web MIXes はプロキシとして動作するが、VPN Gate は VPN サーバとして動作するため、任意の TCP および UDP アプリケーションの通信が可能である。VPN Gate は、Infranet と同様に中継サーバを使用するが、ユーザと中継サーバとの間のトンネル通信が遮断されないようにするために、新たな手法を提案する。VPN Gate は、Telex のように、大量の IP アドレス上で通信を中継する仕組みを利用し、検閲回避を実現する。Telex Station を ISP のインフラに設置してもらう代わりに、VPN 中継サーバをボランティアの PC 上で立ち上げてもらう。

5.1.3　統合的な VPN サーバソフトウェア SoftEther VPN の設計と実装

本研究では、上記のような統合的な VPN サーバソフトウェアを目指して、第 3 章で述べた手法により、SoftEther VPN Server を設計および実装した。

SoftEther VPN Server の設計においては、サポートする VPN プロトコルとして、L2TP/IPsec、SSTP、OpenVPN (L2 および L3)、L2TPv3/IPsec、EtherIP/IPsec および SoftEther VPN Protocol (SEVP) を目標とした。これらのプロトコルをサポートすれば、現在、広く利用されている PC 用 OS である Windows、Mac OS X、Linux、FreeBSD、およびスマートフォン用 OS である iOS、Android、ならびにルータ用 OS である Cisco IOS、IIJ SEIL、NEC IX シリーズ OS からの VPN 接続をサポートできる。

ところが、上記のサポートすべき 7 種類の VPN プロトコル間には、大きな差異がある。そのため、SoftEther VPN Server を統合的な VPN サーバソフトウェアとするためには、これらの差異を吸収する必要があった。これらの VPN プロトコルはレイヤ 3 およびレイヤ 2 に大別することができる。レイヤ 3 VPN プロトコルは IP データグラムの交換を実現し、レイヤ 2 VPN プロトコルはローカルエリアネットワーク (LAN) で使用されている Ethernet のフレームの交換を実現する。この 2 種類の VPN プロトコルは、異なるレイヤのプロトコルの伝送を対象としているので、そのまま相互に接続をすることができない。そこで、本研究では、3 つの手法を検討した。最終的に、VPN サーバ内に、プロトコル間の共通のバスとして、仮想のレイヤ 2 スイッチを作成し、単一のレイヤ 2 スイッチを用いてメッセージを交換する手法を採用した。この手法では、レイヤ 2 フレームはそのまま扱われる一方、レイヤ 3 VPN プロトコルを経由して受信されたすべてのレイヤ 3 パケットは、レイヤ 2 フレームに変換される。送信時は、逆変換される。

レイヤ 3 VPN プロトコルで送受信されるパケットを、レイヤ 2 に変換するために、本

研究では、L2 アダプタを提案した。L2 アダプタは、VPN サーバ内で動作する軽量の IP スタックであり、レイヤ 3 VPN プロトコルを用いて接続中の VPN 接続の数だけ、インスタンスが作成される。L2 アダプタのインスタンスは、レイヤ 2 から見ると、仮想的な NIC に見える。L2 アダプタには、重要な機能として、ARP の処理および DHCP の処理がある。L2 アダプタには、3.3.4 項で述べたように複雑な処理が必要となった。しかし、L2 アダプタの実現によって、SoftEther VPN Server は、レイヤ 2 VPN プロトコルを利用する場合における様々なメリットを存続しつつ、レイヤ 3 VPN プロトコルとの間で、相互接続をすることができるようになった。

SoftEther VPN Server を統合的な VPN サーバソフトウェアとするためには、他にも、色々な設計上および実装上の問題を解決する必要があった。

まず、OS 依存性を少なくし、移植性を向上するために、3.4.5 項で述べたプラットフォーム抽象化レイヤを実装した。プラットフォーム抽象化レイヤは、Windows、Linux、Mac OS X、FreeBSD および Solaris 用に実装されている。Windows とそれ以外の UNIX 互換 OS とは大きく異なるが、UNIX 互換 OS でもそれぞれ細かな違いがある。プラットフォーム抽象化レイヤは、メモリ管理、スレッドおよび同期、ソケット、文字列処理、モノトニック時刻の供給およびデバッグ支援モジュールなどについて、下位で動作する OS の差異を吸収する機能を実現した。

次に、複数の異なる VPN プロトコルをサポートするコードについては、3.4.2 項で述べたように、VPN プロトコルの処理をモジュール化した。これにより、将来あるモジュールでバグが発見されたときに、1 箇所を修正するだけで、そのモジュールを流れるすべての VPN プロトコルに対する処理が修正され、バグ修正の際のコード変更量が削減できる。また、サポートしたい VPN プロトコルが増える場合にも、共通部分が利用できれば、新たに実装すべきコードの量が削減できる。各 VPN プロトコルには、共通のプロトコルを内部利用しているものもある。たとえば、L2TP/IPsec と SSTP は、共に PPP を内部利用している。また、L2TPv3/IPsec はレイヤ 2 VPN プロトコル、L2TP/IPsec はレイヤ 3 VPN プロトコルであるが、L2TP プロトコルの処理は同一である。SoftEther VPN Server の実装において、このような共通部分においては、共通モジュールを実装でき、その上位レイヤおよび下位レイヤを固有モジュールに接続することができるような、モジュール型構造を実現した。モジュール型構造を採用する場合は、モジュール間結合のオーバーヘッドの問題が生じる。そこで、3.4.3 項で述べたような、色々な最適化手法を用いて、パフォーマンスの向上を図った。たとえば、モジュール間のパケットの受渡しは、2 つのモジュールが同一スレッドで動作する場合とマルチスレッドで動作する場合のどち

らでも同様なコードを書くことができるようにし、かつオーバーヘッドを少なくするために、軽量なメッセージキューを実装した。

そして、SoftEther VPN Server は、従来の VPN サーバが外部プログラムの起動を必要としていた PPP 処理や IPsec 処理、パケットフィルタ処理などに必要なモジュールを、VPN サーバソフトウェアのプログラムに内包した。さらに、3.3.5 項で述べたようなユーザモード NAT を実装することにより、SoftEther VPN Server を組み込んだプログラムが一般ユーザ権限で動作している場合でも、VPN に接続したユーザが物理ネットワーク上のホストにアクセスすることが可能になった。SoftEther VPN Server には、マルチテナント機能も実装した。1 つのプロセス内で、複数の分離された VPN を動作させることができるようになった。

3.4.4 項で述べた管理機能により、ユーザ認証やアクセスコントロール、ログ管理などを、複数の異なる VPN プロトコルおよびそのユーザに対して、一括して適用することができるようになった。すべての VPN プロトコルについて、一旦、共通のバスであるレイヤ 2 スイッチを通過させる手法を用いたことにより、これらの管理機能を実装するコードは 1 回のみ記述すれば良くなり、将来的に対応 VPN プロトコルを追加した場合にも、管理設定は自動的に適用されるようになる。なお、3.3.1 項で述べた理由により、本研究の実装では、レイヤ 2 プロトコルは Ethernet のみを想定している。

本研究では、実装した SoftEther VPN Server の評価を行った。多数の VPN プロトコルとの相互運用性の検証については、3.5.1 項で述べたとおり、現在、広く利用されている PC 用 OS である Windows、Mac OS X、Linux、FreeBSD、およびスマートフォン用 OS である iOS、Android、ならびにルータ用 OS である Cisco IOS、IIJ SEIL、NEC IX シリーズ OS からの複数の異なる VPN プロトコルによる VPN 接続が可能であることを検証した。

パフォーマンス評価については、3.5.6 項の結果のとおり、異なる VPN プロトコル間の通信速度について、従来方式と比較して、最大 7% 程度の性能向上を実現した。この結果は、複数 VPN プロトコルをサポートする際、本研究で示した手法が有効であること示している。

安定性については、3.5.7 項のとおり、十分な安定性を有していることを検証した。本研究では、SoftEther VPN Server のバイナリパッケージを 2013 年 3 月にフリーウェアとして公開した。また、2014 年 1 月にソースコードを GNU General Public License (GPL) Version 2 として公開した。SoftEther VPN Server は、2016 年 12 月 20 日までに世界中で 1,009,000 回インストールされた実績を有する。

これらの結果により、本研究の提案手法により、本研究で目標とした統合的な VPN サーバソフトウェアを実現することができることが明らかとなった。

5.1.4 統合的な VPN サーバによる検閲回避 VPN 中継システム VPN Gate の設計と実装

本研究では、統合的 VPN サーバソフトウェアを用いた、検閲回避 VPN 中継システムである VPN Gate の設計および実装を行ない、評価を行った。VPN Gate は、政府のインターネット検閲を回避するための分散型公開 VPN 中継システムであり、多数のボランティアの PC 上に立ち上げられる中継 VPN サーバを世界中に分散配置することにより、検閲への耐性とスケーラビリティの向上を実現した。できるだけ多くのボランティアに協力して立ち上げてもらうためには、簡単なインストールや設定だけで複数の VPN プロトコルに対応した VPN 中継サーバを立ち上げられるようにする必要がある。この際に、本研究における統合的な VPN サーバが利用可能である。さらに、本研究の提案手法である、無実の IP アドレスの混入手法および協調的なスパイ発見手法を実装することで、検閲用ファイアウォールに対する遮断耐性も実現することができる。第 4 章では、これらの設計と実装について述べ、これらの手法を用いた場合の、中国政府の GFW に対する遮断耐性の評価も行った。

中国の Great Firewall (GFW) のような検閲用ファイアウォールの内側のユーザは、検閲用ファイアウォールを回避するため、公開中継サーバを利用することがある。従来の公開中継サーバとしては、公開プロキシ、公開 VPN サーバおよび Tor ノードがあった。しかし、公開中継サーバの IP アドレスのリストを公開すると、検閲当局も同時にこれらの IP アドレスのリストを入手でき、検閲用ファイアウォールの遮断対象とされてしまうという問題があった。さらに、検閲当局はリストを持っていなくても、多数の IP アドレスに対するスキャンを行ない、公開中継サーバを検出すると、自動的にその IP アドレスを遮断リストに追加することができた。

そこで、本研究では、VPN Gate において、検閲用ファイアウォールに対する遮断耐性を向上する新たな手法として、VPN Gate に無実の IP アドレスの混入手法および協調的なスパイ発見手法を提案し、実装した。

無実の IP アドレスの混入手法とは、VPN Gate と無関係な複数の IP アドレスをサーバリストに含める手法である。例えば、極めて重要なサーバ (例: Windows Update のサーバ) を無実の IP アドレスとしてサーバリストに混入する。この手法により、検閲当局は、サーバリスト内のすべての IP アドレスを検閲用ファイアウォールの遮断リストに登録する前に、混入された無実の IP アドレスを識別して取り除く必要が生じる。

協調的なスパイ発見手法とは、すべてのボランティアの VPN 中継サーバが協調動作することにより、スパイコンピュータであると疑われる IP アドレスのリスト (スパイリスト) を生成し、共有することにより、特定の IP アドレスが無実の IP アドレスであるか否かを識別するための調査用通信を発する検閲当局のコンピュータ (スパイコンピュータ) を検出する手法である。各 VPN 中継サーバは、スパイリストに含まれる IP アドレスからのパケットを無視するようになる。その結果、検閲当局は、サーバリストに含まれるある IP アドレスが、本物の VPN 中継サーバであるか、それとも、混入された無実の IP アドレスであるかを見分けることができなくなる。

VPN Gate Server ソフトウェアは、ボランティアが、簡単に VPN Gate Server をインストールし稼働させることができるようにする必要があった。また、Windows、Mac、iPhone、Android などの多様なデバイスから VPN Gate Server に接続したいユーザの需要を満たす必要があった。単一インスタンスによる複数プロトコルへの対応、インストールの容易性、システム管理者権限への非依存、通信ログの統一性および集約性、セキュリティポリシーの設定の一元化などが必要であった。これらの要件の実現のために、本研究で実装した統合的な VPN サーバソフトウェアを利用した。第 4 章で設計、実装した SoftEther VPN Server を VPN Gate Server に組み込むことにより、VPN に関する専門知識を有しないボランティアでも、VPN Gate Server を立ち上げることができるようにした。さらに、VPN Gate のクライアントソフトウェアである VPN Gate Client も、VPN Gate Server として動作するようにした。

このようにして、多数のボランティアの VPN サーバを協調動作させることにより、検閲耐性を向上することができることを示すことができた。

本研究では、統合的な VPN サーバソフトウェアの実現と、検閲耐性を向上させるための各種手法との組み合わせにより、中国の Great Firewall (GFW) のような検閲用ファイアウォールに対する強力な遮断耐性を実現することができた。検閲耐性を向上させるための各種手法を適用することにより、全体の 60% - 70% の VPN サーバが、GFW による遮断を避けることができた。したがって、検閲用ファイアウォールがある国のユーザは、サーバリスト内からランダムに 2 個のサーバを選択すると 84% 以上、ランダムに 3 個のサーバを選択すると 93% 以上の確率で、到達可能な VPN サーバに接続できるようになった。統合的な VPN サーバソフトウェアの寄与により、VPN Gate は多数のボランティアによって維持されるシステムとなった。ボランティアのおかげで、本研究では、中継機能の提供に関するコストをかける必要がない。代わりに、分散配置されたボランティアが、それぞれの PC において、少量の電力および帯域幅を提供することにより、VPN Gate

ネットワークのために貢献している。これと比較して、検閲当局は、複雑な調査用パケットの送付プログラムなどを実装し、これらを継続的に動作させる必要があり、このような検閲用インフラストラクチャの構築と運用には、多額のコストを支出しなければならない。

このように、本研究では、統合的な VPN サーバソフトウェアの実現と、検閲耐性を向上させるための各種手法との組み合わせは、検閲システムと、検閲回避システムとの間のいたちごっこにおいて、検閲回避システムの側をコスト的に有利とする状況を実現することができることを示すことができた。

5.2 研究の発展

本研究において提案した統合的な VPN サーバソフトウェアの実現手法により、従来の VPN サーバソフトウェアの実装において生じていた多くの問題が解決された。しかしながら、未解決の問題も存在する。たとえば、VPN プロトコルには、ロードバランシングに対応したものと、対応していないものとがある。統合的な VPN サーバソフトウェアを用いてロードバランシングを実現しようとしても、対応している VPN プロトコルしかロードバランシングを実現できない。また、プラットフォーム抽象化レイヤにより OS 間の差異を吸収する方法が原因で、特定の OS で一部の機能が利用できない問題がある。たとえば、ユーザモード NAT 機能において、ユーザモードから ICMP パケットを送信することができる OS とできない OS とがある。今後は、このような制限事項を解消していきたい。

本研究において提案した統合的な VPN サーバソフトウェアは、C 言語ランタイムライブラリや暗号化ライブラリ、および OS の提供する API や、OS の有するメモリ管理、スレッド管理、ファイルシステム、TCP および UDP ソケットなどに依存している。これらの依存先の OS 機能は、広く使われている Windows、Linux、Solaris、FreeBSD および Mac OS X で提供されているものであり、今後、これらの OS でこれらの機能が利用できなくなる可能性は低い。しかし、OS のアップデートにより突然 OS 側の実装が変更されることがあり、パフォーマンスに影響が生じたり、OS 側のバグが原因で VPN サーバソフトウェアの動作に不具合が発生したりする可能性が考えられる。例えば、本研究において実装した統合的な VPN サーバソフトウェアの並列処理は、OS のマルチスレッド機能を利用している。そのため、コンテキストスイッチのタイミングなどが OS の実装によって変更されることがある。このため、遅延やジッタの発生度合いが変化することがある。今後は、VPN サーバのようなネットワークソフトウェアが、下位の OS の実装変更による挙動の変化によって受ける影響をさらに少なくする方法について、研究していきたい。

本論文では、統合的な VPN サーバソフトウェアを実装する際に、外部プログラムやラ

イブラリへの依存を少なくすることを実現し、従来の VPN サーバソフトウェアが外部プログラムに依存していた処理を、単一のプログラム内に内包させた。従来手法では、外部プログラムやライブラリ内のコードにバグがある場合は、外部プログラムやライブラリを修正すれば問題が解決できた。しかし、本研究の方式の場合は、VPN サーバソフトウェア側のコードを修正する必要があるというデメリットがある。この問題は、VPN サーバソフトウェアのソースコードをオープンソース化し、第三者が不具合を簡単に修正できるようにすることで解決できる。本研究の実装においても、オープンソースとして公開した後、3.5.5 項で述べたように、多数のコントリビュータからパッチを送付され、不具合の修正のほか、機能面・性能面の向上に踏み込んだ改良が行われた。今後、SoftEther VPN Server のような、多機能なネットワークソフトウェアが、より多数のプログラマにコントリビュートしてもらえるようにするための手法や、プログラム内の適切なモジュール分割手法などについて、研究をしていきたい。

本研究において実装した統合的な VPN サーバソフトウェアが物理ネットワークとの間で通信を行う際には、OS の有するソケット API を利用する。これは、Data Plane Development Kit (DPDK) [41] などを用いて物理ネットワークを直接駆動する方式と比較すると低速である。物理ネットワークを直接駆動し、OS の有するソケット API を利用せずに VPN サーバソフトウェアを実装することができれば、パフォーマンス上のメリットが大きいと考えられる。VPN サーバソフトウェアは、物理ネットワークの駆動、演算、メモリ操作および必要最低限のディスクの読み書き (設定情報の保存) があれば動作するはずである。現在、OS を呼び出す必要がある処理のほとんどは、物理ネットワークを間接的に駆動するためのソケット API の呼び出しである。将来は、システム管理者が VPN サーバの実行時に DPDK を利用するオプションを有効にすることにより、従来の OS のプロトコルスタックに相当する部分について、VPN サーバソフトウェア内部で実装したコードが呼び出され、さらに統合性の高い VPN サーバソフトウェアとして動作するような拡張を行いたい。

本研究で述べたインターネット検閲に対する耐性を向上するための手法は、時間が経過することにより、効果が薄くなると考えられる。たとえば、中国の GFW 管理局は、この論文を読むことにより、本研究による検閲耐性の実現手法を知ることができるため、これに対抗した新たな検閲手法を実装し、GFW に適用をすることができる。そこで、今後も常に GFW のような検閲用ファイアウォールについて研究を行い、また新たな検閲耐性の向上手法を考案し、実装していきたい。

本研究における VPN サーバソフトウェアは、ユーザモードプログラムとして実装した。

3.5.6 項の結果のように、本研究における実装は、従来手法におけるカーネルモード内の実装と比較すると、ユーザ/カーネル間のスイッチングのオーバヘッドにより速度が低下し、不利となる場合があった。この問題を解決するため、多くの部分をカーネルモードで実行可能にする手法が考えられる。そのためには、カーネル内にモジュールをロードし、ネットワークスタックの挙動を動的に変更可能にする必要がある。現在のところ、汎用的な OS 上にはそのような仕組みは存在していない。今後は、OS にそのような機能を実装する方法について検討を行いたい。

謝辞

本研究を行うにあたり、筑波大学の新城靖准教授、加藤和彦教授、板野肯三教授、佐藤聡准教授、中井央准教授には、多くのご助言およびご指導をいただきました。ここに深く感謝申し上げます。特に新城准教授には、本研究期間のみではなく、筑波大学に入学してから現在までの約 14 年間に渡り頻繁に技術的なご助言をいただきましたが、本研究にはそれらを基礎に考えたアイデアによるものが多数含まれています。

本論文をまとめるにあたり、筑波大学の和田耕一教授、李頡教授、面和成准教授には、貴重なアドバイスをいただきました。

京都大学の馬谷誠二助教、Boston University の Sharon Goldberg 准教授には、本研究を発表するにあたり、貴重なアドバイスをいただきました。

Georgia Institute of Technology の Calton Pu 教授には、SoftEther VPN Server の研究について、貴重なアドバイスをいただきました。

東京大学の竹内郁雄教授、石田晴久教授、和田英一教授、品川高廣准教授、筑波大学の白川友紀教授、田中二郎教授、北川高嗣教授、慶應義塾大学の清木康教授、砂原秀樹教授、奈良先端科学技術大学院大学の山口英教授、独立行政法人情報処理推進機構の田代秀一博士、産業技術総合研究所の高木浩光博士、インターネットイニシアティブの歌代和正氏、NTT データの宮本久仁男博士、NTT コミュニケーションズの宮川晋博士には、2003 年以降、SoftEther VPN Server または VPN Gate の研究に関係する貴重なアドバイスをいただきました。

独立行政法人情報処理推進機構および経済産業省の皆様、国立研究開発法人科学技術振興機構の皆様には、未踏ソフトウェア創造事業や戦略的創造研究推進事業およびその他の支援事業を通じて、SoftEther VPN Server の初期のバージョンや VPN 機能の開発およびその後の発展に資する開発資金および各種のご支援をいただきました。国立研究開発法人情報通信研究機構の皆様には、VPN プロトコルモジュールに関する研究を行うため、特別研究員として同機構に所属していた際に、研究環境等のご支援をいただきました。これらのソフトウェアは、SoftEther VPN Server のモジュールの原型となっています。

国立情報学研究所学術基盤課の皆様には、本研究の実験インフラとして必要となる学術情報ネットワーク (SINET) への直接接続を無償で提供していただきました。また、IPTP Networks の Vladimir Kangin 氏には、本研究の実験インフラとして必要となった BGP トランジットを無償で提供していただきました。

警察庁生活安全局情報技術犯罪対策課の皆様および総務省総合通信基盤局データ通信課の皆様には、VPN Gate の公開や運用につきまして、貴重なアドバイスをいただきました。

ぷらっとホーム株式会社の皆様には、SoftEther VPN Server について、各種の開発や実験を行う際の機材を無償で貸与いただきました。

東日本電信電話株式会社の皆様には、本研究を実施するための、NTT ビルを初めとした色々な場所に設置した実験用インフラの構築において大変お世話になりました。

株式会社インターリンクの原口譲治氏を初めとした皆様には、SoftEther VPN Server について、ISP 環境での実験にご協力をいただきました。

UC Davis の小西芽衣博士、豊橋技術科学大学の吉田光男博士、ソフトイーサ株式会社の杉山哲男博士、桑名潤平博士、伊藤隆朗氏、原哲哉氏、松本智氏、畠山元也氏、山本真弓氏、公庄博氏、登義人氏、高槻高等学校の上嶋千春氏、株式会社イーゲルの榮樂英樹氏、アクセンチュア株式会社の久池井淳氏、株式会社グーグルの池嶋俊氏、Increments 株式会社の及川卓也氏、筑波大学の高野和美氏、根本晃輔氏、菊地綾音氏、小西響児氏および Christopher Smith 氏には、SoftEther VPN Server または VPN Gate の研究において様々なご協力をいただきました。

著者が所属している筑波大学ソフトウェア研究室の皆様、実験環境を活用させていただいた筑波大学学術情報メディアセンターの横山憲彦氏、真中剛司氏を初めとする皆様、筑波大学産学リエゾン共同研究センターの皆様、筑波大学先端学際領域研究センターの皆様、筑波大学情報科学類 WORD 編集部室の皆様、筑波大学情報科学類産学官連携推進室の皆様、つくば市役所産業振興センターの皆様には、本研究を進めるにあたって、多大なご支援をいただきました。

SoftEther VPN Server の商用版である PacketiX VPN Server の利用者の皆様や、その他のソフトイーサ社のサービスの利用者の皆様にこれらの製品・サービスをご購入いただいたことにより、本研究を継続するに足る十分な資金を得ることができました。

100 万人を超える SoftEther VPN Server のオープンソース版および VPN Gate の利用者の皆様からのフィードバックにより、本研究の質を高めることができました。

上記を含めた数多くの方々のご協力により、本研究を円滑に実施し、まとめることができたと思います。誠にありがとうございました。

参考文献

[1] D. Anderson: "Splinternet Behind the Great Firewall of China", *ACM Queue*, Vol. 10, No. 11, 2012.

[2] Azureus Software, Inc., "Azureus User Guide", http://azureus.sourceforge.net/.

[3] M. Beck, R. Magnus and U. Kunitz : "Linux Kernel Internals with Cdrom (3rd Edition)", ISBN 0201719754, http://dl.acm.org/citation.cfm?id=560490, September 1, 2002.

[4] O. Berthold, H. Federrath, and S. Koopsell: "Web MIXes: A system for anonymous and unobservable Internet access", *Designing Privacy Enhancing Technologies, Springer LNCS 2009*, pp 115-129, 2001.

[5] L. Bettini: "Implementing Domain-Specific Languages with Xtext and Xtend", Packt Publishing Ltd, 2013.

[6] M. S. Bhiogade: "Secure socket layer.", *Computer Science and Information Technology Education Conference*, pp. 85-90. 2002.

[7] BitTorrent, Inc., "BitTorrent User Manual", http://www.bittorrent.com/help/manual/.

[8] M. Brain: "Network communications using the NetBEUI protocol", *Dr Dobb's Journal-Software Tools for the Professional Programmer 19*, no. 11 , p.82-87, 1994.

[9] S. Canaves: "China's social networking problem", *IEEE Spectrum 48, no. 6*, p.74-77, 2011.

[10] B. Chen and R. Morris: "Flexible Control of Parallelism in a Multiprocessor PC Router", *USENIX Annual Technical Conference*, 2001, pp. 333–346.

[11] S. Cheshire and M. Krochmal: "Multicast DNS", *RFC 6762*, 2013.

[12] J. Claassen, R. Koning, and P. Grosso: "Linux containers networking: Performance and scalability of kernel modules", *Network Operations and Management Symposium (NOMS)*, IEEE/IFIP, pp. 713-717, 2016.

[13] R. Clayton, S. J. Murdoch, and R. N. M. Watson: "Ignoring the Great Firewall of China", *In the Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*, pp.20-35, 2006.

[14] B. Cohen: "Incentives build robustness in Bit-Torrent", *In Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, pp.68-72, 2003.

[15] "DD-WRT". [Online]. Available: https://www.dd-wrt.com/ [Accessed: 30-Nov-2016].

[16] L. Deri and R. Andrews: "N2N: A Layer Two Peer-to-Peer VPN", *Resilient Networks and Services*, pp. 53–64, 2008.

[17] P. Deutsch: "GZIP file format specification version 4.3", *RFC 1952*, 1996.

[18] R. Dingledine, N. Mathewson, and P. Syverson: "Tor: the Second-Generation Onion Router", *the 13th conference on USENIX Security Symposium*, pp. 303–320, 2004.

[19] J. Douceur: "The Sybil Attack", *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, January 2002.

[20] B. T. Doshi, N. Farber, P. Harshavardhana, R. Kapoor, A. Kashper, S. S. Katz, K. S. Meier-Hellstern and T. S. Guiffrida: "ATM network architecture employing an out-of-band signaling network", *U.S. Patent 5,568,475*, October 22, 1996.

[21] R. Droms: "Dynamic Host Configuration Protocol", *RFC 2131*, 1997.

[22] D. E. Eastlake 3rd: "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", *RFC 4305*, 2005.

[23] K. Egevang and P. Francis: "The IP Network Address Translator (NAT)", *RFC 1631*, 1994.

[24] S. Eisenbach, V. Jurisic, and C. Sadler: "Feeling the way through DLL Hell", *Proceedings of the First Workshop on Unanticipated Software Evolution*, Malaga, Spain. 2002.

[25] 榮樂英樹, 新城　靖, 加藤和彦: 「ユーザレベル OS のためのユーザレベルネットワーク機能」, 『情報処理学会 第 3 回情報科学技術フォーラム (FIT 2004)』, B-028, pp.161-162 (2004).

[26] M. Feilner: "OpenVPN: Building and Integrating Virtual Private Networks", *Packt Publishing*, 2006.

[27] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger: "Infranet: Circumventing Web Censorship and Surveillance", *In the Proceedings of the 11th USENIX Security Symposium*, August 2002.

[28] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger: "Thwarting web censorship with untrusted messenger discovery", *In Proceedings of the 3rd Workshop on Privacy Enhancing Technologies (PET 2003)*, Springer LNCS 2760, pp. 125-140, 2003.

[29] R. Fielding, J. Gettys, J. Mogul. H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee: "Hypertext Transfer Protocol - HTTP/1.1", *RFC 2616*, 1999.

[30] R. Fisher: "60 GHz WPAN standardization within IEEE 802.15. 3c", *2007 International Symposium on Signals, Systems and Electronics*, 2007.

[31] D. Flanagan and Y. Matsumoto: "The ruby programming language", O'Reilly Media, Inc", 2008.

[32] M. Gates, A. Warshavsky, J. Dugan, and et al.: "iperf - Perform Network Throughput Tests", http://iperf.sourceforge.net/, 2010.

[33] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn: "Point-to-Point Tunneling Protocol (PPTP)", *RFC 2637*, 1999.

[34] R. Housley and S. Hollenbeck: "EtherIP: Tunneling Ethernet Frames in IP Datagrams", *RFC 3378*, 2002.

[35] P. Howarth, J. Cameron, and M. Gillham: "Poptop - An Open Source Implementation of a PPTP Server", http://sourceforge.net/projects/poptop/.

[36] D. Hartmeier: "Design and performance of the OpenBSD stateful packet filter (pf)", *Proc. of the USENIX Annual Tech. Conf*, pages 171-180, 2002.

[37] C. Huang, D. A. Maltz, J. Li and A. Greenberg: "Public DNS system and global traffic management" , *INFOCOM, 2011 Proceedings IEEE*, pp. 2615-2623, 2011.

[38] G. Huang, S. Beaulieu, and D. Rochefort: "A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers", *RFC 3706*, 2004.

[39] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg: "UDP Encapsulation of IPsec ESP Packets", *RFC 3948*, 2005.

[40] IEEE 802.1Q: IEEE Standards for Local and Metropolitan Area Networks: "Virtual Bridged Local Area Networks", 2003.

[41] Intel: "Data Plane Development Kit". [Online]. Available: http://dpdk.org/ [Ac-

cessed: 30-Dec-2016].

[42] "IronRuby tools for Visual Studio". [Online]. Available: http://ironruby.net/tools/. [Accessed: 30-Nov-2016].

[43] J. Jeong, J. Park H. Kim: "Service discovery based on multicast DNS in IPv6 mobile ad-hoc networks", *Vehicular Technology Conference, VTC 2003-Spring, The 57th IEEE Semiannual*, vol. 3, pp. 1763-1767, 2003.

[44] Katalix Systems Ltd: "The manual page of openl2tpd".

[45] C. Kaufman, P. Hoffman, Y. Nir and P. Eronen: "Internet Key Exchange Protocol Version 2 (IKEv2)", *RFC 5996*, 2010.

[46] S. Kent: "IP Encapsulating Security Payload (ESP)", *RFC 4303*, 2005.

[47] T. Kivinen, A. Huttunen, B. Swander, and V. Volpe: "Negotiation of NAT-Traversal in the IKE", *Internet Drafts draft-ietf-ipsec-natt-ike-08*, 2004.

[48] T. Kivinen, A. Huttunen, B. Swander, and V. Volpe: "Negotiation of NAT-Traversal in the IKE", *RFC 3947*, 2005.

[49] J. Knoble: "Almost Internet with Slirp and PPP", *Linux Journal*, 1996.

[50] S. Kopsell and U. Hilling: "How to achieve blocking resistance for existing systems enabling anonymous web surfing", *In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, 2004.

[51] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley and J. Turner: "Algorithms to accelerate multiple regular expressions matching for deep packet inspection", *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 339-350, 2006.

[52] L. Lamport: "Concurrent reading and writing of clocks", *ACM Transactions on Computer Systems (TOCS) 8*, no. 4, p.305-310, 1990.

[53] J. Lau, M. Townsley, and I. Goyret: "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", *RFC 3931*, 2005.

[54] H. K. Le, D. Henriksson and T. Abdelzaher: "A practical multi-channel media access control protocol for wireless sensor networks", *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 70-81. IEEE Computer Society, 2008.

[55] S. J. Leffler and M. K. McKusick: "The Design and Implementation of the 4.3 Bsd Unix Operating System", ISBN 0201546299, March 1991.

[56] W. Louati, B. Jouaber, and D. Zeghlache: "Configurable Softwarebased Edge

Router Architecture", *Computer Communications 28 (14)*, pp. 1692–1699, 2005.

[57] G. Lowe, P. Winters and M. L. Marcus: "The great DNS wall of China", *MS, New York University 21*, 2007.

[58] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell and C. Wright: "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks", RFC 7348, 2014.

[59] K. Mandke, S. H. Choi, G. Kim, R. Grant, R.C. Daniels,W. Kim, R.W. Jr. Heath, and M.S. Nettles: "Early results on Hydra: A Flexible MAC/PHY Multihop Testbed", *Vehicular Technology Conference*, pp. 1896–1900, 2007.

[60] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert and V. Paxson: "China' s great cannon", *Citizen Lab, University of Toronto, Technical Report*, 2015.

[61] D. Martin and A. Schulman: "Deanonymizing users of the SafeWeb anonymizing service", *Proceedings of the 11th USENIX Security Symposium*, 2002.

[62] U. Maurer: "Modelling a public-key infrastructure", *European Symposium on Research in Computer Security*, pp. 325-350. Springer Berlin Heidelberg, 1996.

[63] D. McCoy and J. A. Morales and K. Levchenko: "Proximax: A measurement based system for proxies dissemination," *Financial Cryptography and Data Security*, 2011.

[64] D. Merkel: "Docker: lightweight linux containers for consistent development and deployment", *Linux Journal 2014(239)*, p.2, 2014.

[65] Microsoft: "NET_BUFFER Architecture (Windows Drivers)", http://msdn.microsoft.com/en-us/library/windows/hardware/ff568377(v=vs.85).aspx, November 2012.

[66] Microsoft: "Secure Socket Tunneling Protocol (SSTP)", http://msdn.microsoft.com/en-us/library/cc247338.aspx, 2013.

[67] Microsoft: "Unencapsulated MS-CHAP v2 Authentication Could Allow Information Disclosure", https://technet.microsoft.com/library/security/2743314.aspx, 2012.

[68] Microsoft: "Visual Studio Uninstaller". [Online]. Available: https://github.com/Microsoft/VisualStudioUninstaller [Accessed: 12-Dec-2016].

[69] S. C. Mitchell and J. S. Quarterman: "Using ARP to Implement Transparent

Subnet Gateways", *RFC 1027*, 1987.

[70] J. Mirkovic and P. Reiher: "A taxonomy of DDoS attack and DDoS defense mechanisms", *ACM SIGCOMM Computer Communication Review 34, no. 2*, p. 39-53, 2004.

[71] K. Miyazawa, S. Sakane, K. Kamada, M. Kanda, and A. Fukumoto: "Design and Implementation to Support Multiple Key Exchange Protocols for IPsec", *Linux Symposium*, p.143. 2006.

[72] A. Mockus, R. T. Fielding and J. Herbsleb: "A case study of open source software development: the Apache server", *Proceedings of the 22nd international conference on Software engineering*, pp. 263-272, 2000.

[73] R. Morris, E. Kohler, J. Jannotti, and F. Kaashoek: "The Click Modular Router", *the seventeenth ACM symposium on Operating systems principles*, pp. 217–231, 1999.

[74] G. C. Murphy, M. Kersten, and L. Findlater: "How are Java software developers using the Elipse IDE?" IEEE software 23.4, p.76-83, 2006.

[75] R. Nee and R. Prasad: "OFDM for wireless multimedia communications", Artech House, Inc., 2000.

[76] 登 大遊: 「SoftEther の内部構造」, 『情報処理学会誌「情報処理」45 (10)』, pp. 1057–1062, 2004.

[77] 登 大遊, 新城 靖, 佐藤 聡: 「SoftEther VPN Server: マルチプロトコル対応のクロスプラットフォームなオープンソース VPN サーバ」, 『ソフトウェア科学会 コンピュータソフトウェア』, Vol.32, No.4, pp.3-30, 2015.

[78] D. Nobori and Y. Shinjo: "VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls", *11th USENIX NSDI*, pp. 229–241, 2014.

[79] E. Nygren, R. K. Sitaraman and J. Sun: "The Akamai network: a platform for high-performance internet applications", *ACM SIGOPS Operating Systems Review 44, no. 3*, p.2-19, 2010.

[80] OpenVPN Community: "OpenVPN Community Software", http://openvpn.net/index.php/open-source/overview.html.

[81] "OpenWRT". [Online]. Available: https://openwrt.org/ [Accessed: 30-Nov-2016].

[82] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bolcskei: "An overview of MIMO

communications-a key to gigabit wireless", *Proceedings of the IEEE 92*, no. 2, 198-218, 2004.

[83] H. G. Perros: "Connection-oriented networks: SONET/SDH, ATM, MPLS and optical networks", John Wiley & Sons, 2005.

[84] D. C. Plummer: "An Ethernet Address Resolution Protocol", *RFC 826*, 1982.

[85] J. Postel: "INTERNET CONTROL MESSAGE PROTOCOL", *RFC 792*, 1981.

[86] G. N. Purdy: "Linux iptables pocket reference", O' Reilly, 2004.

[87] "rbox-tor: an easy to use Tor server" , redct. [Online]. Available: http://redct.info/rbox/tor.html. [Accessed: 13-Sep-2013].

[88] C. Rigney, S. Willens, A. Rubens and W. Simpson: "Remote Authentication Dial In User Service (RADIUS)", *RFC 2138*, 2000.

[89] Y. Sato and Y. Hamazaki: "DeleGate: A general purpose application level gateway", *Worldwide Computing and Its Applications*, pp. 426-441, 1997.

[90] G. S. Sidhu, R. F. Andrews and A. B. Oppenheimer: "Inside AppleTalk", Addison-Wesley, 1990.

[91] W. Simpson: "The Point-to-Point Protocol (PPP)", *RFC 1661*, 1994

[92] "strongSwan". [Online]. Available: https://www.strongswan.org/ [Accessed: 12-Dec-2016].

[93] N. Tolia, M. Kaminsky, D. G. Andersen and S. Patil: "An Architecture for Internet Data Transfer.", *In NSDI*, 2006.

[94] "Tor Project: obfsproxy", torproject.org. [Online]. Available: https://www.torproject.org/projects/obfsproxy. [Accessed: 04-Sep-2013].

[95] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter: "Layer Two Tunneling Protocol 'L2TP'" , *RFC 2661*, 1999.

[96] "UberCloud". Available: https://www.theubercloud.com/ [Accessed: 24-Dec-2016].

[97] "Visual Studio". [Online]. Available: https://www.visualstudio.com/. [Accessed: 30-Nov-2016].

[98] "Free VPN Info and PC Tips @ VpnSurfing.com", vpnsurfing.com. [Online]. Available: http://www.vpnsurfing.com/. [Accessed: 04-Sep-2013].

[99] "VyOS". [Online]. Available: https://vyos.io/ [Accessed: 30-Nov-2016].

[100] L. Wang and J. Kangasharju: "Real-world sybil attacks in BitTorrent mainline

DHT", 2012 IEEE Global Communications Conference (GLOBECOM), pp. 826-832, 2012.

[101] T. Wilde: "Great Firewall Tor probing Circa 09 Dec 2011" [Online]. Available: https://gist.github.com/da3c7a9af01d74cd7de7. [Accessed: 04-Sep-2014].

[102] S. Williams and C. Kindel: "The component object model: A technical overview", *Dr. Dobbs Journal*, p.356-375, 1994.

[103] P. Winter and J. R. Crandall: "The Great Firewall of China", 2012. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:610844/FULLTEXT01.pdf [Accessed: 12-Dec-2016].

[104] P. Winter and S. Lindskog: "How the great firewall of china is blocking Tor," *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[105] E. Wustrow, S. Wolchok, I. Goldberg, and J. Alex Halderman: "Telex: Anti-censorship in the network infrastructure", *In the Proceedings of the 20th USENIX Security Symposium*, August 2011.

[106] D. I. Wolinsky, K. Lee, P. O. Boykin, and R. Figueiredo: "On the Design of Autonomic, Decentralized VPNs", *Collaborative Computing: Networking, Applications and Worksharing*, 2010.

[107] X. Xu, Z. M. Mao and J. A. Halderman: "Internet censorship in China: Where does the filtering occur?", *In Proceedings of the 12th International Conference on Passive and Active Measurement (PAM 11)*, pp. 133-142, 2011.

[108] C. V. Zhou, C. Leckie, and S. Karunasekera: "A survey of coordinated attacks and collaborative intrusion detection", *Computers & Security*, Vol.29, No.1, pp.124-140. 2010.

[109] "VMware Virtual Appliance Marketplace". [Online]. Available: https://solutionexchange.vmware.com/store/category_groups/virtual-appliances/ [Accessed: 24-Dec-2016].

[110] Y. T. Yiton: "ZLIB and related programs for beam dynamics studies", *AIP Conference Proceedings*, pp. 279-279, 1993.

[111] P. Chandra, M. Messier and J. Viega: "Network security with OpenSSL", O' Reily, June 2002.

著者の論文リスト（添付）

公表論文リスト

1. 登 大遊, 新城 靖, 佐藤 聡: "SoftEther VPN Server: マルチプロトコル対応のクロスプラットフォームなオープンソース VPN サーバ", ソフトウェア科学会 コンピュータソフトウェア, Vol.32, No.4, pp.3-30 (2015).

2. Daiyuu Nobori and Yasushi Shinjo: "VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls", 11th USENIX Symposium on Networked Systems Design and Implementation, pp.229-241 (2014).

その他

1. Yasushi Shinjo, Naoki Kainuma, Daiyuu Nobori, Akira Sato: "Magic Mantle using Social VPNs against Centralized Social Networking Services", IEEE 14th Annual Conference on Privacy, Security and Trust (PST), 8 pages, (2016).

2. Yasushi Shinjo, Xiao Kunyao, Naoki Kainuma, Daiyuu Nobori, and Akira Sato: "Friend News System: A Modern Implementation of Usenet over Social VPNs", 7th IEEE International Conference on Social Computing and Networking, pp.432-440 (2014).

3. 海沼 直紀, 新城 靖, 登 大遊, 肖 焜瑶, 佐藤 聡, 中井 央: "プライバシを保護するための VPN を用いた ソーシャルアプリケーション実行環境", 情報処理学会第 26 回コンピュータシステム・シンポジウム (ComSys2014), pp.3-15 (2014).

4. 海沼 直紀、新城 靖、登 大遊、櫻井 孝一、佐藤 聡、中井 央: "SoftEther VPN を用いたソーシャルアプリケーション実行環境の構築", 情報処理学会第 25 回コンピュータシステム・シンポジウム (ComSys2013), ポスターセッション, 2 pages (2013).

5. Younggyun Koh, Calton Pu, Yasushi Shinjo, Hideki Eiraku, Go Saito, Daiyuu Nobori: "Improving Virtualized Windows Network Performance by Delegating Network Processing", The 8th IEEE International Symposium on Network Computing and Applications (IEEE NCA09), pp.203-210 (2009).

6. 登 大遊, 加藤 和彦, 新城 靖, 板野 肯三, 佐藤 聡, 中井 央: "PC 上で動作するスケーラブルな IP ネットワーク実験システム", 第 70 回情報処理学会全国大会講演論文集, 1Y-8 (2008).

7. 登 大遊: "SoftEther VPN の内部構造", 情報処理学会 第 17 回コンピュータシステムシンポジウム論文集, 6 pages (2005).

8. 登 大遊: "SoftEther の内部構造", 情報処理学会誌「情報処理」, Vol.45, No.10, pp.1057-1062 (2004).

9. 登 大遊: "SoftEther による Ethernet の仮想化とトンネリング通信", 情報処理学会 第 45 回プログラミングシンポジウム論文誌 pp.147-158 (2004).