平成13年度学士論文

特定ユーザーのみが利用可能な
仮想プライベート・ネットワーク

東京工業大学 理学部 情報科学科
学籍番号 9804020
宇崎 央泰

指導教官
千葉 滋 講師

平成14 年2 月22 日

概要

盗聴や改ざんなどの危険などのあるネットワークをあたかも専用線のように利用できる技術が、仮想プライベートネットワーク（Virtual Private Network：VPN）である。この VPN を利用することで、コストのかかる専用線を利用しプライベートなネットワークを構築する代わりに、インターネットを経由してプライベートなネットワークを構築することができ、柔軟で安価なプライベートネットワークが構築できるようになる。しかし、従来の VPN にはユーザーの概念が取り入れられていない。ユーザーがホストや VPN 機器の間に VPN を構築するときにはホストの認証を行うが、その後は認証を行わない。そのため、共有計算機などから VPN を構築すると他のユーザーもその VPN を使えてしまという問題が起きる。

そこで、本研究では、VPN にユーザーの概念を取り入れ、複数のユーザーが存在するホスト上でも特定のユーザーのみが利用できるパーソナル VPN を提案する。パーソナル VPN とは、VPN を構築するときにユーザー認証を行うことができる VPN である。さらに、VPN を構築した後も、その VPN を利用するときはユーザーの認証を行う。このような認証を行うことにより、特定のユーザーのみが VPN を構築することができ、またその VPN を他のユーザーは利用することができなくなる。このようなパーソナル VPN をカーネルに手を加えることをせずに Linux 上に実装した。VPN の構築には SSL を利用し、その VPN に IP パケットを流すために Divert ソケットを利用した。また、IP パケットを VPN に流す前に、proc ファイルシステムを利用しユーザー ID をチェックすることでユーザー認証を行った。

謝辞

本論文を作成するにあたり、欠点を的確に指摘し、指導していただいた指導教官の千葉滋講師に感謝致します。

また、研究について数々の助言をしていただいた東京大学の光来健一氏、励ましの言葉を下さった研究室の皆様に感謝致します。

目 次

図 目 次

表 目 次

第1章　はじめに

企業などが、自分たちの支店と本店のネットワーク同士を接続しようとする場合、専用線を利用するのが一般的であった。インターネットなどのオープンネットワークを経由して接続すると、途中で盗聴や改竄などの危険があるため、インターネットを経由した接続形態はほとんど取られていなかった。

しかし、暗号化、ハッシュ関数、認証などの技術が向上したことにより、盗聴や改竄などの危険を防ぐことができるようになった。これらの技術を利用して、オープンネットワークを仮想的に専用線のように使うことができるようにする技術およびこの技術を用いて構築したネットワークを、仮想プライベートネットワーク（Virtual Private Network：VPN）と呼ぶ。このVPNを利用することで、インターネットのようなオープンネットワークを経由してネットワークを構築した場合でも、盗聴や改竄の危険を少なくできる。専用線を引く変わりにインターネットとVPNを利用することで、コストの削減ができる。また、専用線を利用してネットワークを構築した場合、各ネットワークの位置が固定されてしまうのに対し、インターネットとVPNを利用した場合は、インターネットに接続できる環境があればよいので、各ネットワークの位置は自由に配置を変える事ができる。

VPNを利用する場合、ネットワーク同士を接続するだけでなく、ホストがVPNの機能を備えていれば、ネットワークとホストの間、ホストとホストの間でVPNを構築することもできる。たとえば、IPsecやPPTPなどを使い、リモートホストから自分の所属する組織のネットワークに参加する方法は企業などがよく利用している。また、SSHとポートフォワーディングを利用して、一種のVPNを構築することが可能であるが、Windowsでも利用することができるのでよく使われている。しかし、これら既存のVPNの場合、ユーザーの概念が十分に取り入れられていないので、VPNを構築するとそのホスト上のすべてのユーザーがVPNを利用できてしまう。自分専用の計算機からVPNを構築する場合は問題はない

が、多数のユーザーが利用する共用計算機から VPN を構築した場合に、他のユーザーに VPN を利用されるという問題が起きる。たとえば、IPsec ではホストが組織のネットワークに参加するときにユーザー認証を行うことが可能だが、ネットワークに参加後はユーザー認証は行われなないため、他のユーザーも VPN を利用できる。PPTP もネットワークに参加するときに PPP によるユーザー認証が行われるが、その後はユーザー認証は行われないため、他のユーザーも VPN を利用できる。SSH は暗号化通信路を確立するときにユーザー認証を行うが、その後はどのユーザーも SSH のポートフォワーディング機能を利用することができる。

そこで、本研究では、VPN にユーザーの概念を取り入れた、パーソナル VPN を提案する。パーソナル VPN とは、IP パケットごとにユーザー認証を行う VPN である。まず、VPN を構築するときに RSA ユーザー認証を行い、VPN を構築した後は、その VPN を通過する IP パケットの送信ユーザーを調べ、VPN を構築したユーザーだけが利用できるようにする。ユーザーごとに異なる暗号化通信路を用意することで、他のユーザーからの盗聴などを防ぐ。また、このパーソナル VPN 同士を連結する機能をもたせることで、ファイアーウォールなどのために直接通信できないホスト同士の間にパーソナル VPN を構築することができる。

このようなパーソナル VPN をカーネルに手を加えることをせずに Linux 上に実装した。VPN の構築には SSL を利用した。また、VPN 構築のとき、SSL によるサーバーホストの認証を行った。その VPN に IP パケットを流すために、divert ソケットを使い IP パケットをインターセプトした。また、IP パケットを VPN に流す前に、proc ファイルシステムを利用しユーザー ID をチェックすることでユーザー認証を行った。ユーザー認証に失敗した IP パケットは、そのまま IP 層にもどすので、通常のルーティングが行われる。

今回実装したパーソナル VPN を通した場合の通信のスループットと平均レスポンスタイムを計測した。ベンチマークソフトには WebStone を使った。通常の通信のスループット、平均レスポンスタイムと比較して、約 10 分の 1 から数百分の 1 程度であった。ただし、プログラムに多少問題があり、パケットの再送が頻発することがありパフォーマンスが低下することがあったので、改善の余地はある。

以下、2 章では VPN を支える技術と現在実現されている代表的な VPN について述べる。3 章では本研究で提案するパーソナル VPN の設計と現在の実装状況について述べる。4 章ではパーソナル VPN を利用した場合

のスループットと平均レスポンスタイムを計測する実験を行い、その結果について考察する。最後に5章で本論文をまとめる。

第2章　関連研究

2.1　仮想プライベートネットワークとは

仮想プライベートネットワーク（Virtual Private Network：VPN）とは、盗聴、改竄、なりすましなどの危険があるネットワークを、暗号化、一方向ハッシュ関数、認証などの技術を使うことで、プライベートなネットワークを仮想的に実現したネットワークである。この VPN を利用することで、インターネットなどの安全でないネットワークをあたかも専用線であるかのように利用することができる。しかし、VPN では帯域などの通信品質が確保されないため、専用線のような高い通信品質を得ることができないといった問題がある。しかし、目的に応じて使えば、コスト面や接続地点を自由に変更できるなど拡張性の高さを生かした利用法を実現できる。

2.2　仮想プライベートネットワークの利用形態

仮想プライベートネットワークの利用形態は、以下の３つが考えられる。

1. ネットワーク間接続 VPN 離れたネットワーク同士をインターネットを経由して接続する方法である。各ネットワークのインターネットへの出入り口に VPN サーバーを置く (図 2.1)。その VPN サーバーが暗号化などをおこなうので、ネットワークの各端末は VPN の仕組みを持つ必要はない。

2. 端末－ネットワーク間での VPN 端末とネットワークを結ぶ形態で、特にリモートアクセスなどの目的で利用される。例えば自分がいつも持ち歩くノートパソコンなどに、対応する VPN ソフトウェアをあらかじめインストールしておけば、外出先からインターネット経由で安全に内部のサーバにアクセスすることができる (図 2.2)。

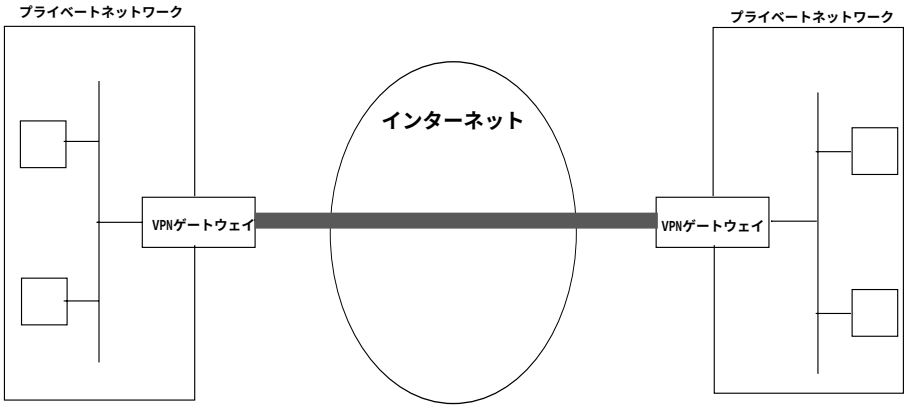

図 2.1: ネットワーク間 VPN

3. 端末ー端末間での VPN 端末と端末との間で VPN を利用する方法である (図 2.3)。この形態では、End-to-End のセキュリティを確保できる。

2.3　仮想プライベートネットワークを実現する技術

2.3.1　暗号化技術

暗号化とはあるアルゴリズムを使用しデータを加工し、それを解く鍵が無ければ解読出来ない（復号化できない）ようにすることである。暗号化により盗聴の危険を防ぐことができる。VPN で使われる暗号化技術は、主に次の二つが利用される。

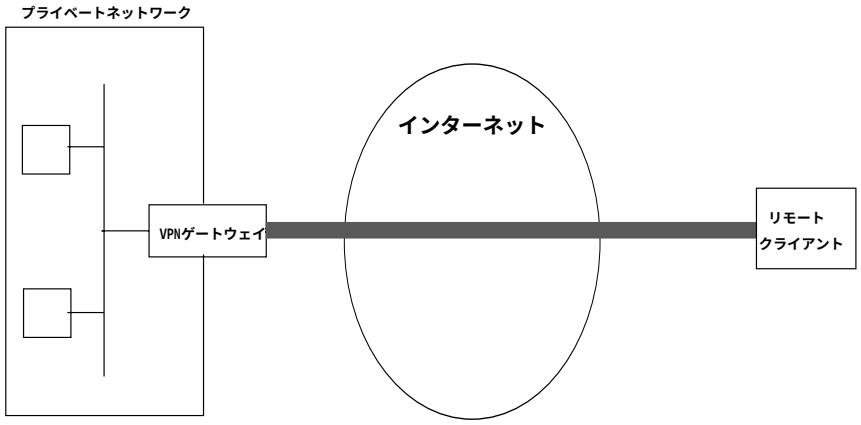

図 2.2: 端末－ネットワーク間 VPN

共有鍵暗号方式

暗号化と復号化に同じ鍵を使う暗号方式を共有鍵暗号方式という（秘密鍵暗号化方式と呼ばれることもある）。

共有鍵暗号の問題点

1. 鍵の配送問題：どうやって送信者と受信者が鍵を共有するか。暗号化は安全な送信路がないときに利用したいのに、鍵を共有するためには安全な送信路がないといけないという矛盾に陥る。

2. 鍵の管理：N 人の利用者と暗号化通信をする場合、各自 (N-1) 個の鍵を管理しなくてはいけない。

公開鍵暗号方式

公開鍵暗号方式では、二つの鍵を生成する。通常、一方の鍵を公開し、もう一方の鍵を秘密にする。前者を公開鍵、後者を秘密鍵と呼ぶ。公開鍵と秘密鍵は１対１に対応しており、公開鍵から秘密鍵を類推することは困難である。

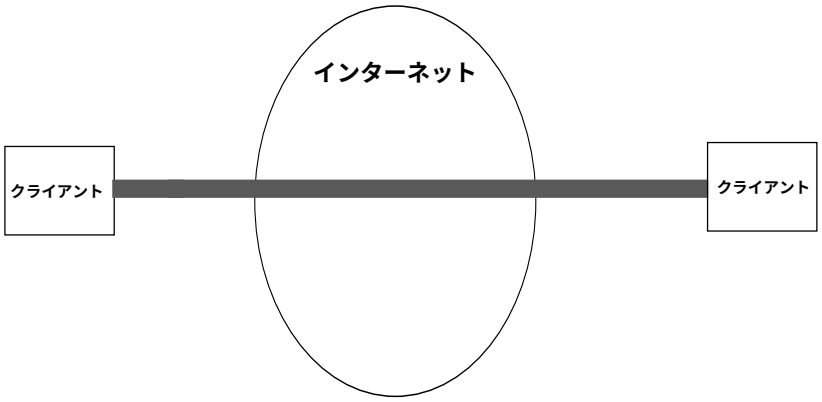

図 2.3: 端末－端末間 VPN

公開鍵暗号の特徴

1. 鍵の秘密配送が不要
   共有鍵暗号では、暗号通信に先だって秘密に共有鍵を配送（共有）する必要があるが、公開鍵暗号では、公開されている相手の公開鍵を入手するだけでよい。

2. 秘密に保持する鍵が少ない
   N 人の利用者と暗号通信する場合、共通鍵暗号では、N 個の異なる共通鍵を秘密に保持する必要があるが、公開鍵暗号では、自分の秘密鍵を保持するだけでよい。

3. 電子署名（デジタル署名）が実現できる
   共通鍵暗号では、ある共通鍵で暗号化された文書がその共通鍵を共有する当事者のいずれによって暗号化されたかが特定できないが、公開鍵暗号では、ある秘密鍵で暗号化された文書はその秘密鍵を保持する当人しか暗号化できないため誰が作成したかが特定できる。

4. 共通鍵暗号に比べて処理が低速
   公開鍵暗号では、共有鍵暗号に比較して、暗号処理が低速である。このため、多量のデータの暗号通信には共通鍵暗号を利用し、共通

鍵暗号に用いられる共通鍵の秘密配送と電子署名に公開鍵暗号を利用するのが一般的である。

5. 公開鍵の正当性確保が必要
公開鍵暗号では、公開鍵を公開できる反面、それが偽の公開鍵にすり変えられると大きな問題となる。これを防ぐため、公開鍵が誰の公開鍵であるかを証明する機関（認証局 (Certification Authority：CA)）[1]を利用するのが一般的である。

6. 暗号通信による相手認証は不可能（匿名暗号通信）
共通鍵暗号では、当事者しか知らない共通鍵を用いて暗号通信するため、暗号文が正しく復号できるかどうかによって通信相手を認証できる。しかし公開鍵暗号では公開鍵を用いて誰でも暗号通信できるため、受け手は誰から暗号文が送られてきたかが分からない。(なお公開鍵暗号の電子署名機能を用いれば、相手認証が可能である)。

共通鍵を公開鍵暗号方式で暗号化して送信し、それ以後の通信はその共通鍵で暗号化するという手法が一般的に使われている。具体的な暗号化の方式としては、RSA（Rivest Shamir Adelman）がある。

2.3.2　一方向ハッシュ関数

一方向ハッシュ関数
ハッシュ関数とは、任意長のデータ x から、固定長のデータ h(x) に変換する関数のことである。さらに、次の特徴を持つ関数を一方向ハッシュ関数という。

1. h(x) から x を逆算できない。

2. x が与えられたとき、 h(x) = h(y) となる y を求めることが困難である。

一方向ハッシュ関数による改竄検出
一方向ハッシュ関数は、入力のメッセージを 1 文字でも変えると、その値から計算したハッシュ値がまったく異なった値となる性質を持っている。

[1]verisign など

そのため、データの改竄の有無をチェックするために、使用することができる。具体的には、メッセージの送信者が、メッセージとそのハッシュ値を一緒に送信し、受信者は、受け取ったメッセージを同じハッシュ関数でハッシュ値を計算して、両方のハッシュ値の比較により改竄の有無を調べる。ハッシュ値の偽造を防ぐために、送信者は自分の秘密鍵でそのハッシュ値を暗号化して送信するか、共有鍵の情報をメッセージとして含むハッシュ値を送信する。

一方向ハッシュ関数の例

一方向ハッシュ関数の例としては、SHA、MD5 などがある。

2.3.3　認証技術

他人が本人の振りをしてネットワークを不正利用する「なりすまし」と呼ばれる問題を防ぐために、アクセスしている者が本人であるかどうかを判断するために必要な技術が認証である。

送信側と受信側の暗号化鍵（復号化鍵）または認証鍵が一致しないとデータの復号化やメッセージ認証に失敗し、送られてきた情報を受け取らないようにすることで、鍵を所有しない第三者のなりすましを防ぐ。

2.4　VPN を実現するプロトコル

VPN を構成する主なプロトコルをＯＳＩ参照モデルのレイヤ別に図 2.4 で示す。

2.4.1　IPsec

IPsec[1] は IP レベルでの認証および暗号化を行う VPN である。認証のために認証ヘッダ（Authentication Header）[4] を使用する（改定後の IPsec[5] では暗号ペイロードにもパケット認証の機能がある）。この認証ヘッダは、IP パケット全体の完全性を保証する (図 2.5)。MD5 や SHA などの一方向性ハッシュ関数を IP パケット全体に適用し、その結果を認証データとする。このため、IP アドレスを含む IP ヘッダの完全性が保証されることになり、ホスト認証が可能となる。

図 2.4: レイヤ別の暗号化・認証プロトコル

ネットワークとホストの間やホスト同士の間で VPN を構築する場合、不特定多数のユーザーからアクセスがあるので、VPN を構築する前にユーザー認証を行う必要がある。この場合、VPN の構築は特定のユーザーしかできない。しかし、VPN 構築後はホスト上の他のユーザーも VPN を利用できる。

認証ヘッダの構文 (図 2.6 参照)

1. ペイロード長（Length)
   32 ビットを単位とする

2. 予約（RESERVED）
   16 ビット全てを 0 にセット

3. 通し番号（Sequence Number Field）
   リプレイ攻撃を防止するためのもの RFC 1826 には定義されていない

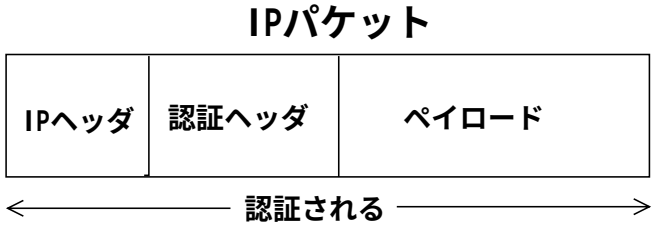

図 2.5: IPsec により認証される範囲

Next Header	Length	RESERVED
Security Parameters Index		
Sequence Number Field		
Authentication Data (variable number of 32-bit words)		

図 2.6: 認証ヘッダ

4. セキュリティ・パラメータ・インデックス（SPI:Security Parameters Index）
   セキュリティ・アソシエーションを特定する
   セキュリティ・アソシエーションには、暗号化や認証にどのアルゴリズムを使用しているかを決定する情報が含まれている。

5. 認証データ（Authentication Data）
   一方向ハッシュ関数を用いたチェックサムの計算結果。
   チェックサムは、ペイロードのデータ、IP ヘッダのフィールド、およびセキュリティ・アソシエーションで合意されている秘密情報を使って計算される。

認証データの長さは、チェックサムを計算するために選ばれたアルゴリズムに依存する。
受信者は、パケットの内容と SPI が示す秘密情報を使ってチェックサムを計算し、その結果をパケットに格納されている認証データと比較する。

2.4.2　データリンク層のプロトコル

データリンク層に位置して、Point-to-Point Protocol(PPP) プロトコルを IP パケットにカプセル化して送る VPN プロトコルとして、Point-to-Point Tunneling Protocol(PPTP）[3] と Layer 2 Forwarding(L2F）[10] がある。これらのプロトコルによって、リモートアクセスサーバーが、加入電話/ISDN 経由のアクセスに対する認証のみではなく、インターネットを経由したエンドユーザーの認証も可能となった。PPP 上で利用できる IP、AppleTalk、IPX/SPX などで、これらのプロトコルを利用できる。また、PPTP と L2F を統合した Layer 2 Tunneling Protocol(L2TP)[9] というプロトコルがある。いずれのプロトコルも VPN を確立するときにユーザー認証を行うが、VPN 確立後は他のユーザーからも利用できる VPN になってしまう。

Point-to-Point Tunneling Protocol(PPTP)

PPTF は、米マイクロソフトが提案している IP トンネリング関連のセキュリティ・プロトコルである。データを IP カプセル化すると同時に、付加した IP ヘッダーにユーザー認証用のデータを埋め込んでやり取りすることで、インターネットを介して接続する場合でもエンド・エンドでユーザー認証を実現できる。

Layer 2 Forwarding(L2F)

L2F は、米シスコ・システムズ社が提案したトンネリング・プロトコルである。次の三段階で認証を行う。最初の段階では、ISP が PAP、CHAP、EAP 等を使ってユーザーを認証し、ユーザーが認証されると VPN トンネルを構築する。VPN トンネルが構築されると、ISP のアクセスサーバが認証情報を VPN サーバへと転送し、ネットワークサーバが、「接続要

求コール」を受けるかどうかを判断することによって 2 番目の認証フェーズを引き受ける。「接続要求コール」が受け付けられれば、オプションでネットワークサーバは 3 番目の認証フェーズを開始することができる。このステップではトンネルは送信メディアとしか扱われず、RAS（リモートアクセスサーバ）がアクセスユーザーを認証するのとほとんど同じような形でユーザーを認証する。

Layer 2 Tunneling Protocol(L2TP)

L2TP は、ダイヤルアップ通信で用いられる PPP 通信をトンネリングするためのデータリンク層のプロトコルである。L2TP ではリモートユーザーは端末からアクセスポイントにダイヤルアップ PPP 接続を行なう。着信を受けた VPN 装置 (LAC：L2TP Access Concentrator) は PAP や CHAP などの PPP のユーザー認証によって、代理的に正当なユーザーであることを確認した後、アクセス側の VPN 装置 (L2TP Network Server) と通信してトンネルを生成する。トンネルが確立された後は、ダイヤルインした内部の認証サーバ (RADIUS など) による正式な認証を経て、PPP のネゴシエーションを行なう。

VPN での認証はトンネルを確立するための認証と、ユーザー・アカウント認証の 2 つのフェイズを指す。前者のトンネルを確立するための認証は ISP 内に置かれ、登録ユーザーかどうかの判断とトンネルを張るための情報をアクセスサーバに提供する役割を持っている (TMS:Tunnel Management Server)。一方、PPTP や L2TP などのトンネリングプロトコルのユーザー認証には PAP、CHAP、MS-CHAP、EAP などの PPP 認証が用いられる。また、RADIUS サーバなどと組み合わせることで単なるユーザー認証だけでなく、アクセス制御と統計情報の収集などが行なえる。

2.4.3　Secure Shell(SSH)

アプリケーション層に位置して、rlogin などのリモート系のコマンド（ネットワークを介して別の計算機にログインしたり、遠隔地のマシンでコマンドを実行したり、他のマシンへファイルを移動 (コピー) したりする）アプリケーションに対してセキュアな通信を提供するのが、SSH である。リモート系のコマンドアプリケーションとしては、rlogin 以外にも、

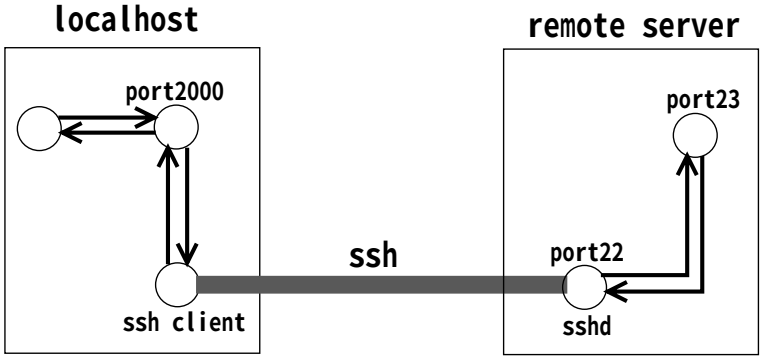

図 2.7: ポートフォワーディングの例

rsh (リモートシェル)、rcp (リモートファイルコピー)、rdist などのアプリケーション全てに適用される。 SSH は、UNIX 系の OS ばかりではなく、WINDOWS にも適用できる。

SSH にはポートフォワーディングと組み合わせて使用することができる。ポートフォワーディングにより任意の TCP ポートを転送し、その通信路を暗号化するため、一種の VPN を構築することができる。localhost の 2000 番ポートを remote server の 23 番 (telnet) ポートへ転送したときの例を図 2.7 に示す。しかし、この SSH を経由するポートフォワーディングを利用するときに、ユーザー認証は行われない。したがって、他のユーザーも SSH 経由のポートフォワーディングを利用できてしまう。

第3章　システムの設計と実装

従来の仮想プライベートネットワークはユーザーの概念が取り入れられていなかったために、あるホストのユーザーがVPNを構築すると、そのホスト上の他のユーザーもそのVPNをつかえてしまうという不都合が起きる。特に、共有計算機のような多数のユーザーが同時に利用するようなホストからVPNを構築する場合に問題となる。

本研究で提案するパーソナルVPNは、特定ユーザーのみ利用可能なVPNである。このパーソナルVPNを実現する手順を簡単に述べる。VPNの構築にはSSLを利用する。このとき、SSLネゴシエーションでサーバーホストの認証を行う。サーバーホストの認証は、VPNを構築しようとするホストが詐称されないようにするために行う。サーバー側ではRSAユーザー認証を行うので、SSLネゴシエーションでクライアントホストの認証は行わない。SSLによるVPNが構築後は、divertソケットを利用してIPパケットをインターセプトし、IPパケットのユーザー認証を行い、そのVPNにIPパケットを流す。IPパケットのユーザー認証は、procファイルシステムを利用し、ユーザーID調べることで行う。IPパケットのインターセプトと認証を行い、パーソナルVPNにフォワードするプロセスをパーソナルVPNフォワーダ（PVPNF)と呼ぶことにする。VPNを通してIPパケットを受け取ったサーバー側のPVPNFは、rawソケットを利用してIPパケットをサーバーに送信する。パーソナルVPNでは、PVPNFは信頼できるプロセスであるということが前提となる。

3.1　パーソナルVPN

パーソナルVPNは、あるユーザーが構築したVPNは他のユーザーが利用することはできない。したがって、複数のユーザーが同時に利用している計算機から、自分以外はアクセスを許可したくないホストとの間にVPNを構築しても、自分以外のユーザーがパーソナルVPNを通してそのホストにアクセスすることはできない。たとえばeduにログインし、

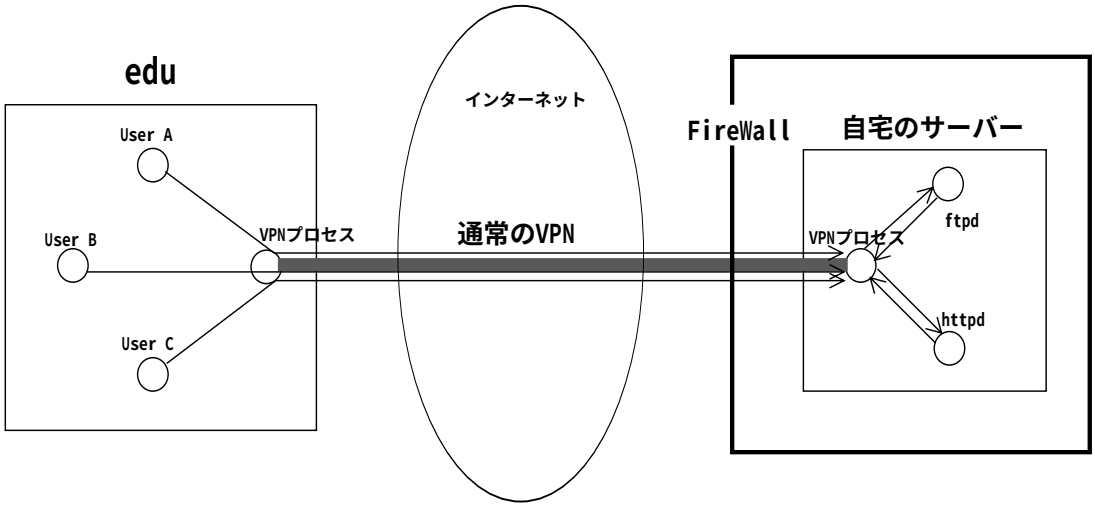

図 3.1: すべてのユーザーが VPN を利用できる

edu から自宅のサーバーとの間に VPN を構築しようとする場合、従来の VPN では、VPN 構築後は edu 上の他のユーザーもその VPN を利用できてしまうため、他のユーザーにも自宅のサーバーにアクセスを許してしまう (図 3.1)。しかし、パーソナル VPN では、VPN 構築後も VPN を利用するときにユーザー認証が行われるため、edu 上の他のユーザーがその VPN を利用することはできず、自宅のサーバーにアクセスされることはない (図 3.2)。

3.2　ユーザーごとの VPN の作成

Secure Socket Layer(SSL) を利用し、ユーザーごとの VPN を作成する。SSL とは Netscape 社により提案された暗号化通信の手法である。SSL はセッション層レベルで暗号化を実現するプロトコルであるが、暗号化だけではなく、相互認証の機能とデータの完全性を保証する MediaAuthenticationCode（MAC) の機能を備えているので VPN として利用することができる。今回は、SSL によるホスト認証は、サーバーホストの認証の

みを行う。サーバー側は RSA ユーザー認証を行うので、クライアントホストを認証する必要はない。さらに、各ユーザーごとに SSL セッションを用意し、ユーザーごとに異なる暗号化を行うことで、ユーザーごとの VPN を作成する。SSL セッションは、SSL のハンドシェイクプロトコルによって作成され、一連の暗号化セキュリティパラメータを定義する。複数の接続がこれらのパラメータを共有することも可能である。SSL セッションを共有、再利用することによって、接続ごとに新しいセキュリティパラメータを指定する必要がなくなり、セッションの確立にかかる時間は短縮できる。

現在は、ユーザーごとにセッションを作成する部分の実装はできていない。したがって、マルチユーザーには対応しておらず、同時に複数のユーザーがパーソナル VPN を構築することはできない。

3.3　IP パケットのインターセプト

サーバー宛ての IP パケットをパーソナル VPN を通して送信するために、IP パケットを IP 層からインターセプトし、カプセル化を行い、VPN に通す必要がある。IP パケットのインターセプトは divert ソケットを利用して実現した。diver ソケットを利用すると、IP 層に入ってくる IP パケットをインターセプトしたりインジェクトすることができる (図 3.3)。

1. divert というプロトコルで raw ソケットをつくる
2. ソケットアドレスをバインドする。ソケットアドレスにはポート番号 4000 を指定する。
3. recvfrom 関数を使い、IP パケットをインターセプトする
4. インターセプトした IP パケットを作ったユーザーのユーザー ID をチェックする (ユーザー ID のチェック方法はユーザー認証の節で述べる)。
5. その IP パケットがパーソナル VPN を構築したユーザーからのものであるとき、IP パケットをプロセス間通信を利用し VPN を構築したプロセスに渡す。VPN を構築したプロセスは、受け取った IP パケットを SSL による VPN に流す。

6. パーソナル VPN を構築したユーザーの IP パケットではないとき、IP パケットは IP 層に戻す。

3.3.1　divert ソケット

divert ソケットを利用すると、ファイアウォールのパケットフィルタで指定した、特定のパケットをインターセプトしユーザー空間に移動することができる。たとえば、172.16.128.10 宛ての IP パケットをインターセプトするならば、次のように ipchains の設定をする（divert ソケットはポート番号 4000 にバインドしておく）。

```
ipchains -A output -s 172.16.128.10 -j DIVERT 4000
```

divert ソケットを作ったプログラムをポート番号 4000 で動かす。その中で、recvfrom() 関数を使うとインターセプトした IP パケットを取得できる。また、インターセプトした IP パケットは、sendto() 関数で再び IP 層に戻すことができる。

divert ソケットを利用することで、Linux のカーネルを改造せずに IP パケットのインターセプト行うことができる。ただし、インターセプトした IP パケットはカーネル空間からユーザー空間へコピーされるので、その分オーバーヘッドが生じる。

Netlink ソケット

Netlink ソケットは、divert ソケットと同様に、ファイアウォールのパケットフィルタリングを利用して IP パケットをインターセプトする。divert ソケットと大きく異なる点は次の 2 つである。

1. Netlink ソケットはポート番号を持たないので、複数のプロセスが Netilnk ソケットを利用する場合、それぞれの Netlink ソケットが異なる IP パケットをインターセプトを行うことは難しい。

2. 外へでていく IP パケットをインターセプトした場合、その IP パケットを IP 層に戻す方法がない。これは Netlink ソケットは、自分で IP

層に戻した IP パケットを再びインターセプトしてしまうからである。

パーソナル VPN では、インターセプトした IP パケットがユーザー認証に失敗した場合、その IP パケットを再び IP 層に戻すようにしている。Netlink ソケットでは IP 層に IP パケットを戻すことはできないため、利用しなかった。

RAW ソケット

RAW ソケットは IP パケットをインターセプトすることはできない。単に IP パケットのコピーが渡されるだけなので、IP パケットが IP 層以外の層に渡されるのを防ぐことはできない。トラフィックの解析などによく用いられる。RAW ソケットを作るときに、指定したプロトコル番号の IP パケットをフィルタリングすることができる。RAW ソケットはファイアウォールの機能を利用しない。

libpcap

libpcap は tcpdump で使われている。libpcap はインターフェースにやってくるトラフィックを listen できる（ppp、ethernet などどんなものでも）。ethernet の場合、NIC をプロミスカスモードにすることで、同じセグメントに属するパケットを解析することができる。パケットのインターセプトやインジェクトはできない。

3.4　ユーザー認証

3.4.1　サーバー側でのユーザー認証

サーバー側では VPN を構築するときに RSA ユーザー認証により、パーソナル VPN を構築しようとしているユーザーの認証を行う。RSA 認証は以下の手順で行われる。

1. VPN クライアントプロセスがこれから VPN を構築するユーザー名と公開鍵を 、VPN サーバープロセスに送る。

2. VPN サーバープロセスは送られてきたユーザー名から、そのユーザーの公開鍵が VPN サーバープロセスのあるディレクトリに存在するか調べる。

3. VPN サーバープロセスは、乱数を生成し、VPN を構築しようとするユーザーの公開鍵で暗号化し、VPN クライアントプロセスへ送る。

4. VPN クライアントプロセスは、受け取ったデータを秘密鍵で復号化し、MD5 を取った結果を秘密鍵で暗号化し VPN サーバープロセスに送る。

5. VPN サーバープロセスは、受け取ったデータを公開鍵で復号化し、これと自分が送ったデータの MD5 を取ったものとを比較し、一致した場合はユーザーが認証される。

3.4.2　パーソナル VPN の入り口でのユーザー認証

パーソナル VPN は、特定ユーザー以外は利用できないようにしなければならない。そのためには、あるユーザーが作成したパーソナル VPN に流す IP パケットのユーザー認証を行う必要がある。

この IP パケットのユーザー認証は proc ファイルシステムを利用して行う。例えば tcp 通信の場合であれば、IP パケットの送信元ポート番号と /proc/net/tcp の local_address のポート番号が一致するエントリのユーザー ID を調べることで、その IP パケットを作ったユーザーを調べることができる。パーソナル VPN を作ったユーザーのユーザー ID と一致した場合は IP パケットをパーソナル VPN に通す。

あるユーザーがパーソナル VPN を構築したときに、PVPNF はそのユーザーのユーザー ID とサーバーの IP アドレスとポート番号を記録しておく。その後クライアントホストからサーバー宛ての IP パケットが発生したとき、divert ソケットと ipchains を利用してその IP パケットをインターセプトする。そして以下の手順で IP パケットのユーザー認証を行う（tcp 通信の場合）。

1. /proc/net/tcp を検索して、インターセプトした IP パケットを作ったユーザーのユーザー ID を調べる。その IP パケットの送信元ポート番号と /proc/net/tcp ファイルの local_address のポート番号が一致するエントリーのユーザー ID がその IP パケットを作ったユーザーのユーザー ID である。/proc/net/tcp の例を以下に示す。

```
sl  local_address rem_address   ...   uid  ...
 0: A9287083:0017 9A287083:0488 ...     0  ...
 1: 00000000:0403 00000000:0000 ...     0  ...
 2: 0100007F:1630 0100007F:0402 ...     1  ...
```

2. そのユーザー ID がパーソナル VPN 構築のときに記録しおいたユーザー ID と一致する場合、IP パケットをパーソナル VPN に流す。一致しない場合は IP パケットは IP 層に戻す。

3.5　パーソナル VPN の連結

ファイアウォールなどにより、パーソナル VPN を構築したいホスト同士が直接通信できない場合がある。このような場合でも、２つのパーソナル VPN でルーティングができるようにし、途中のゲートウェイなどがパーソナル VPN を連結することで、直接通信できないホスト同士の間にパーソナル VPN を構築することができる (図 3.7)。

現在の所、パーソナル VPN の連結機能の実装はできていない。ゲートウェイとの間にパーソナル VPN を構築し、その後はゲートウェイ上の VPN サーバープロセスが raw ソケットを使い IP パケットを送信するようになっているため、直接通信できないホストとも通信することは可能である。ただし、ゲートウェイから先のネットワーク内部では、通信は暗号化などは行わない通常の通信になってしまう (図 3.8)。また、サーバーからの応答パケットはパーソナル VPN を構築したゲートウェイを通過しなければならないという制約がある。もしゲートウェイが複数あり、他のゲートウェイに応答パケットが流れてしまうと、応答パケットはパーソナル VPN を通らずに届いてしまう (図 3.9)。

ゲートウェイから raw ソケットを使って送信するときに、送信元 IP アドレスをゲートウェイの IP アドレスに変換するという方法も考えられるが、そうするとゲートウェイ宛ての IP パケットのうち、どの IP パケットをパーソナル VPN に流すのかが問題となる。そのため、この方法は使わなかった。

3.6　応答パケットの判別

パーソナル VPN を通ってきた通信に対する応答の IP パケットの判別方法は、IP アドレスとポート番号によって見分ける方法を使っている。送信元ポート番号がサーバーがバインドしているポート番号であり、パーソナル VPN のクライアントホスト宛ての IP アドレスのパケットは、パーソナル VPN を通すことにしている。この方法では、あるホストから複数のユーザーが同一ホストとの間にそれぞれパーソナル VPN を構築した場合、応答パケットをとのユーザーのパーソナル VPN に流したらよいか、区別がつかなくなってしまうという問題が起きる (図 3.10)。

サーバーが発信元となり、パーソナル VPN のクライアントホストとの間で通信を行おうとした場合、IP パケットがパーソナル VPN を流れてしまうが、通信ができなくなってしまうことはない。パーソナル VPN を通過した IP パケットは、受信側で raw ソケットにより再び送信される。そのため、IP パケットを受信したクライアント側のプロセスからは、サーバーが直接送信したかのようにみえるからである。

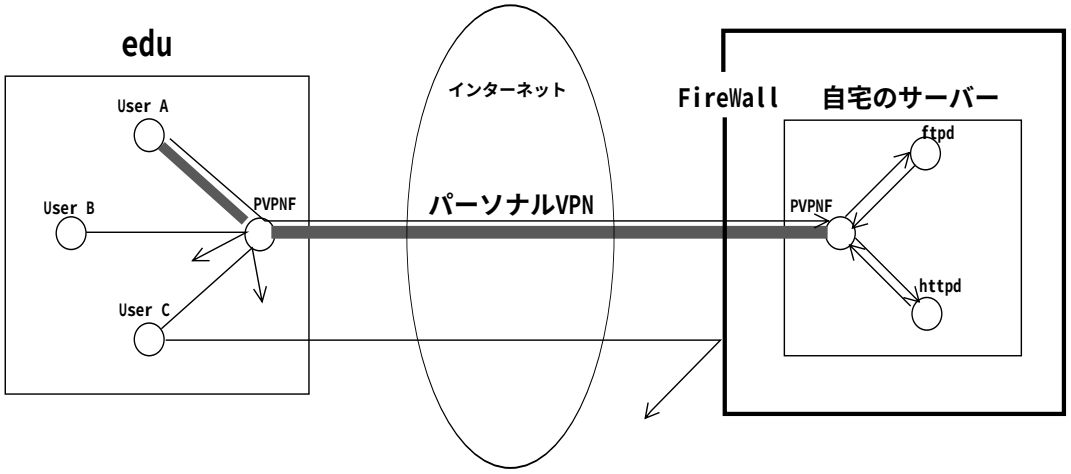

図 3.2: 他のユーザーは VPN を利用できない

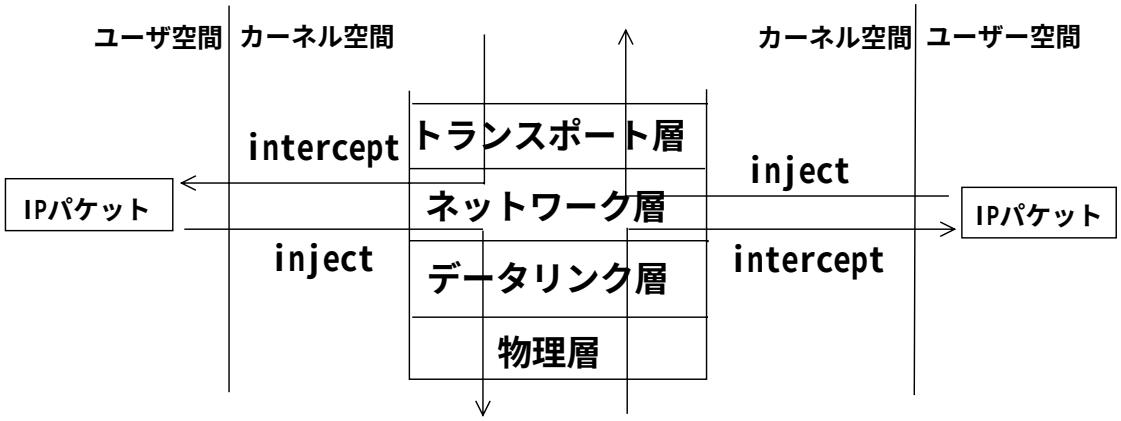

図 3.3: IP パケットのインターセプト、インジェクト

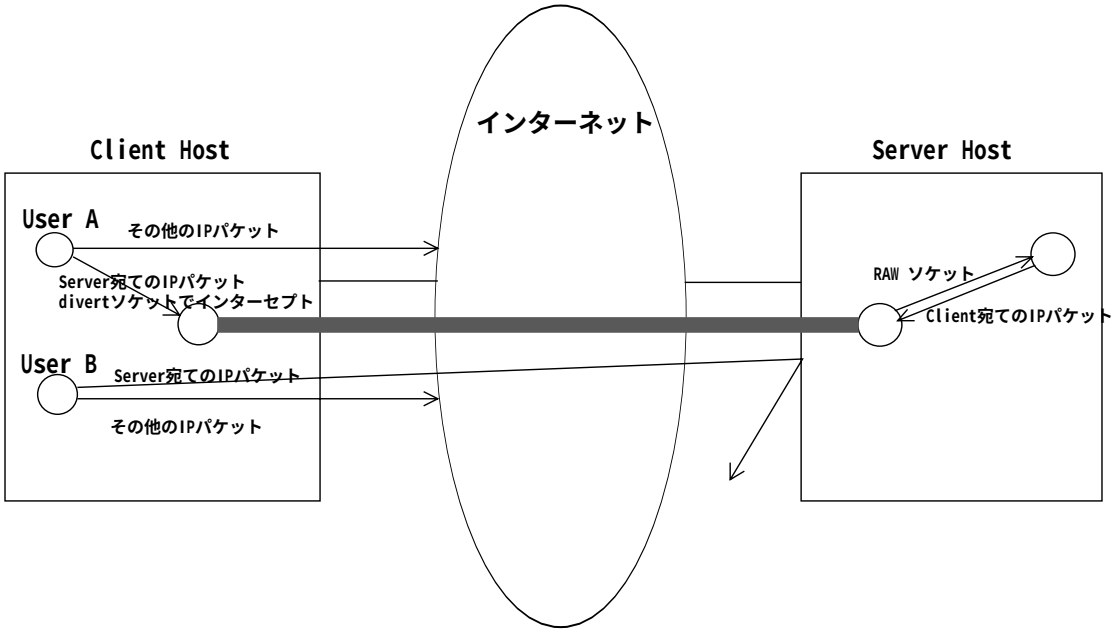

図 3.4: インターセプトした IP パケットの流れ

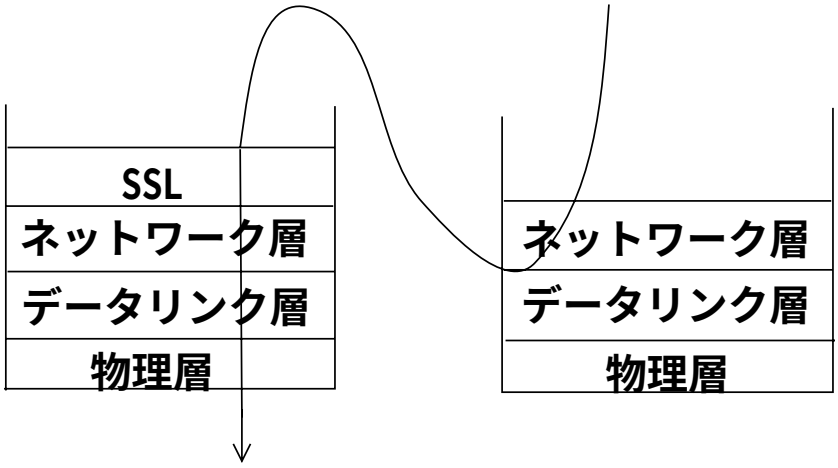

図 3.5: OSI 参照モデルでのパケットの流れ

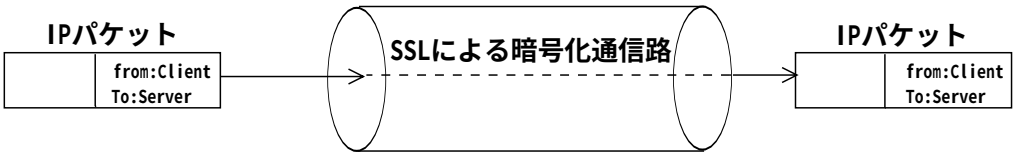

図 3.6: SSL による VPN に IP パケットを流す

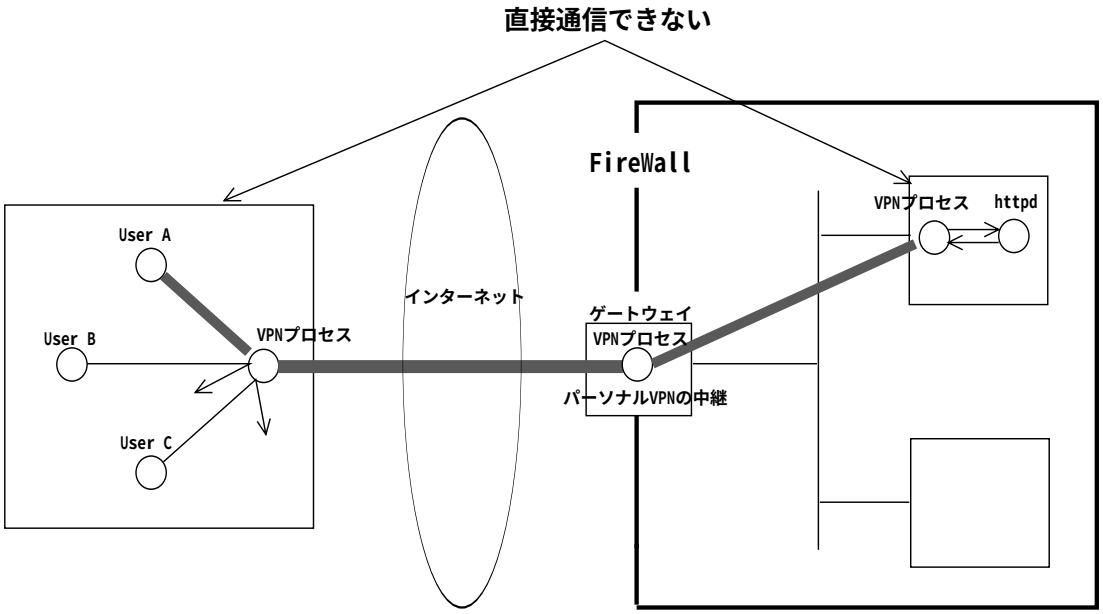

図 3.7: 直接通信できないホスト間にパーソナル VPN を構築する

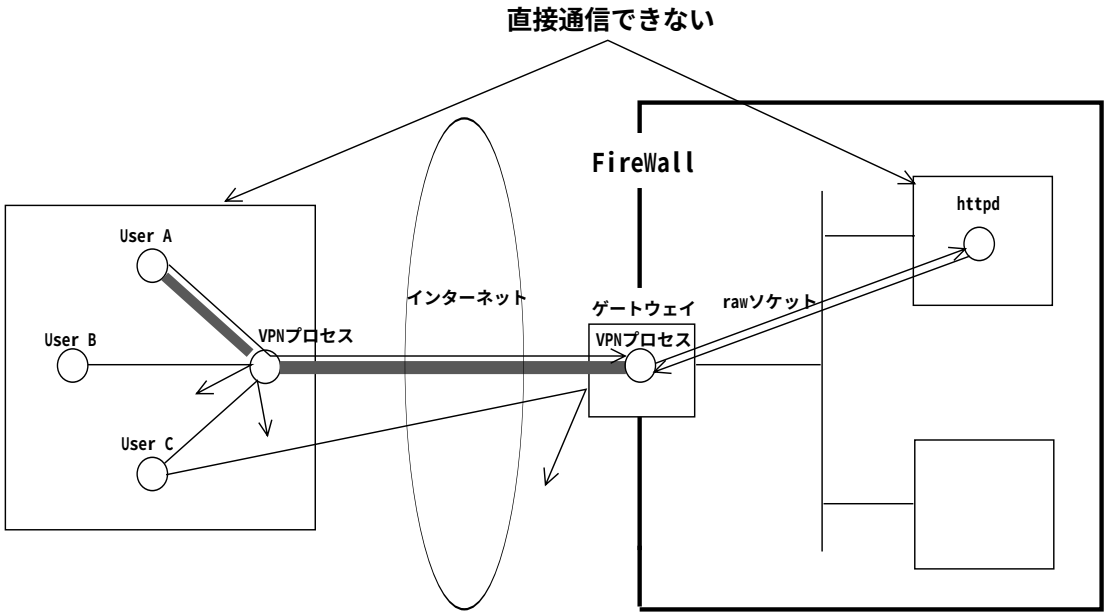

図 3.8: ゲートウェイから IP パケットを送信

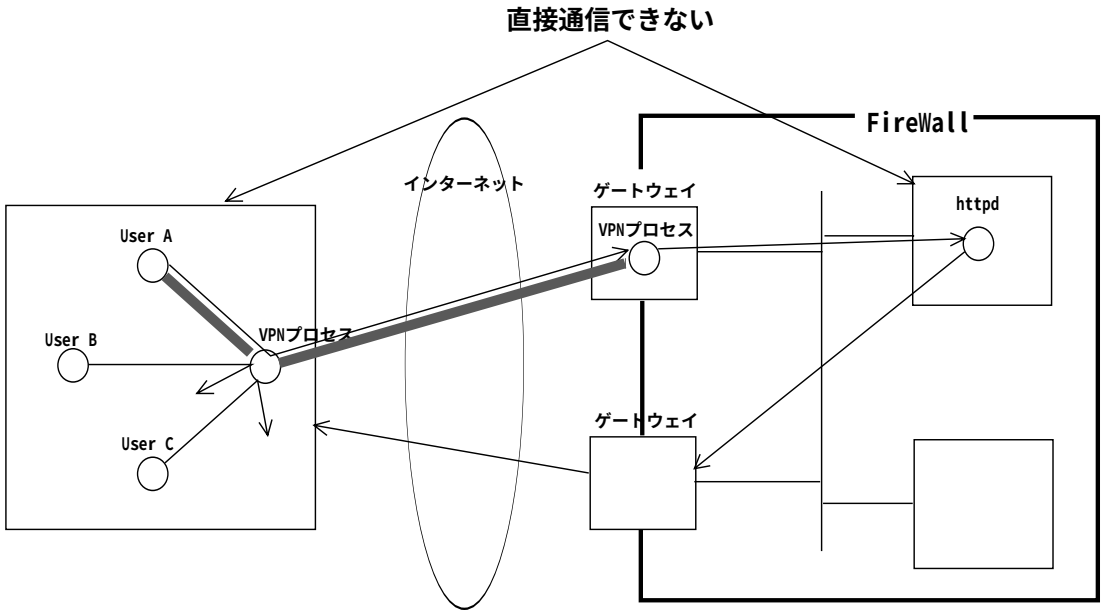

図 3.9: 応答パケットが他のゲートウェイに行く

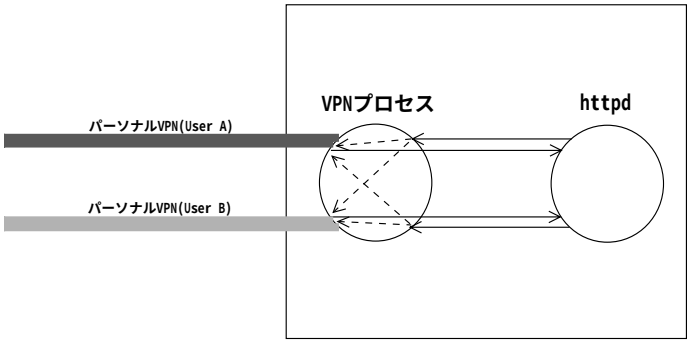

図 3.10: 応答パケットをどの PVPN に流したらよいかわからなくなる問題

第4章　実験

実装によるオーバーヘッドを調べる実験を行った。実装のなかでは、以下にのふたつのオーバーヘッドが大きいと考えた。

1. カーネル空間とユーザー空間との間で IP パケットを移動するときのオーバーヘッド
2. 暗号化を行うことによるオーバーヘッド

このふたつのオーバーヘッドを調べるために、次のような実験を行った。

1. 扱う IP パケットのサイズが小さいものが多い状態から、大きいサイズの IP パケットが増えていくようにして、パフォーマンスの変化を調べる。
2. 暗号化を行う場合と行わない場合の比較をする

4.1　実験環境

実験は以下の環境で行った。

1. クライアントマシン：Pentium　400、Memory384Mbyte
2. サーバーマシン：AthlonXP1800+、Memory640Mbyte
3. クライアントマシンとサーバーマシンは 100Mbit/s の LAN で繋がっている。
4. SSL ライプラリは OpenSSL0.9.6、SSL のバージョンは SSLv3 を用いた。
5. ベンチマークソフトは WebStone を使用した。

表 4.1: Throughput(bytes/sec)

file size(Kbyte)	0	1	10	100	1000
normal	167,724	711,306	3,168,808	8,385,680	8,497,332
noencrypt	1,556	6,137	262,611	650,646	821,635
encrypt	431	2,192	30,791	263,691	629,131

WebStone とは web サーバーの性能測定ツールである。
GPL の元に Mindcraft 社のホームページ (http://www.mindcraft.com/) にて配布されている。

4.2　実験内容

クライアントマシンからサーバーマシンの web サーバーにアクセスをしたときのスループットおよび平均レスポンスタイムを、以下の 3 つの場合について計測した。

1. クライアントマシンから直接サーバーマシンの web サーバーにアクセスする。
2. パーソナル VPN を通してアクセスする。データの暗号化は行わない。
3. パーソナル VPN を通してアクセスする。データの暗号化を行う。

WebStone がベンチマークを計測するのに利用するファイルのファイルサイズは 0byte、1Kbyte、10Kbyte、100Kbyte、1Mbyte の 5 種類のサイズを使用した。

4.3　実験結果

実験結果を表 4.1 と表 4.2 に示す。
表 4.1、 4.2 をグラフにしたものが、図 4.1 と図 4.2 である。

表 4.2: AverageResponseTime(sec)

file size(Kbyte)	0	1	10	100	1000
normal	0.001906	0.001883	0.003298	0.012225	0.123190
noencrypt	0.210098	0.220387	0.040224	0.157616	1.270646
encrypt	0.757750	0.610793	0.342316	0.388082	1.651290

4.4　考察

実験を行う以前は、扱うデータサイズが大きくなり、やりとりする IP パケットのサイズが大きくなるほど、パーソナル VPN のスループットが低下すると予想していた。それは、以下 3 つのことが起きるため、扱う IP パケットのサイズが大きいほどオーバーヘッドが大きくなると考えたからである。

1. divert ソケットを使って IP 層から IP パケットをインターセプトする。このとき、IP パケットはカーネル空間から、ユーザー空間に移動する。

2. カプセル化を解除して取り出した IP パケットを、raw ソケットを使い送り出す。

   以上の二つでは、カーネル空間からユーザー空間、ユーザー空間からカーネル空間への IP パケットの移動の増加によるオーバーヘッドが生じる。当然、移動するデータのサイズが大きいほどオーバーヘッドも大きくなる。

3. インターセプトした IP パケットを、プロセス間通信により VPN プロセスに渡すことによるオーバーヘッド。こちらも、扱うデータのサイズが大きいとオーバーヘッドが増加する。

今回実装したプログラムでは、クライアントからサーバーにデータが届くまでに、カーネル空間とユーザー空間との間でのデータの移動が 6 回、プロセス間通信が 1 回起きることになる (図 4.3 参照)。

それに対して、クライアントとサーバーが普通に通信を行った場合、カーネル空間とユーザー空間との間でのデータの移動は 2 回起こるだけ

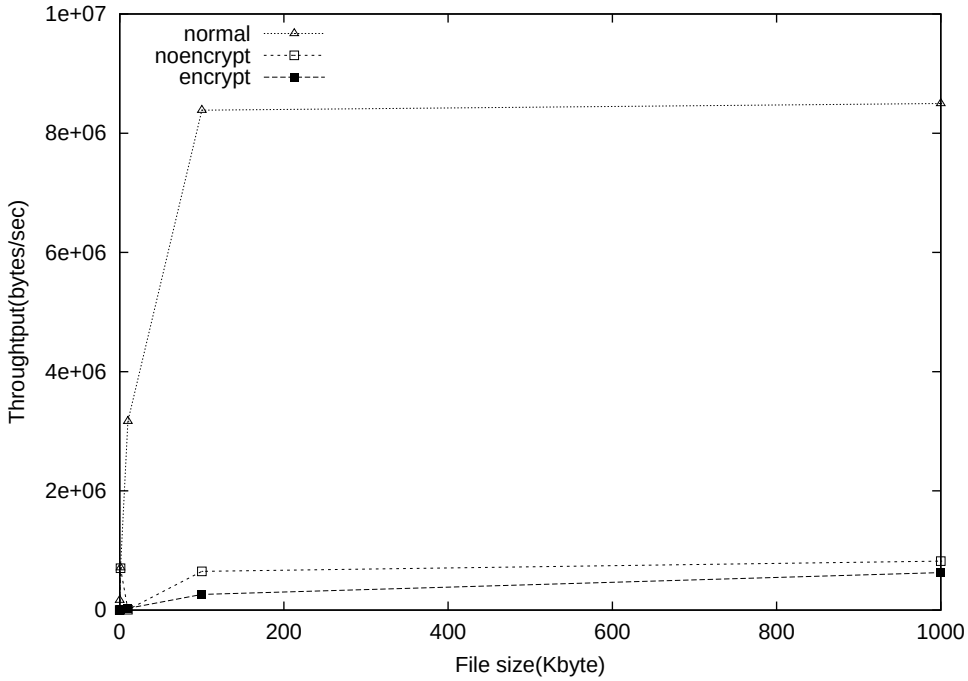

図 4.1: ThroughtpuTime

である。(図 4.4)。

しかし、実験結果からはデータサイズが大きくなるにつれ、普通に通信を行った場合との差が小さくなっている。この原因を調べるためネットワークのトラフィックを調べた。その結果、今回の実装によるパーソナル VPN を通して通信を行っているとき、扱うデータのサイズが小さいときは、タイムアウトによる IP パケットの再送が頻繁に発生している事がわかった (図 4.5 参照)。データサイズを大きくとると、再送はほとんど発生しなかったため (図 4.6 参照)、データのサイズが小さいときよりパフォーマンスが良かったようである。データサイズが小さいときにタイムアウトによる再送が頻発するのは実装に問題があると思うのだが、原因を究明している時間がなかったので今後の課題としたい。

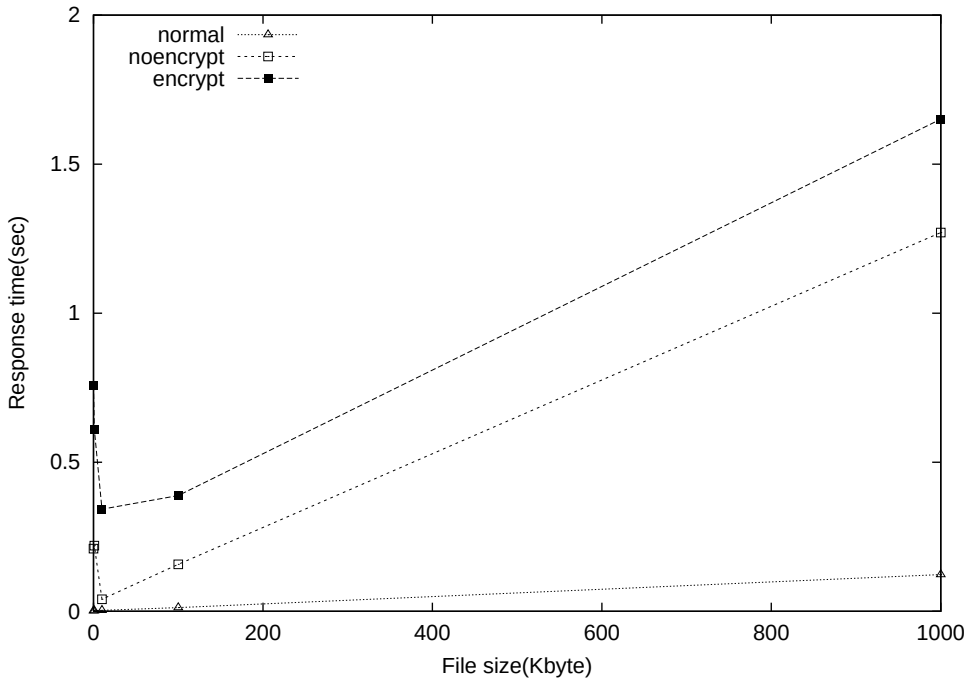

図 4.2: ResponseTime

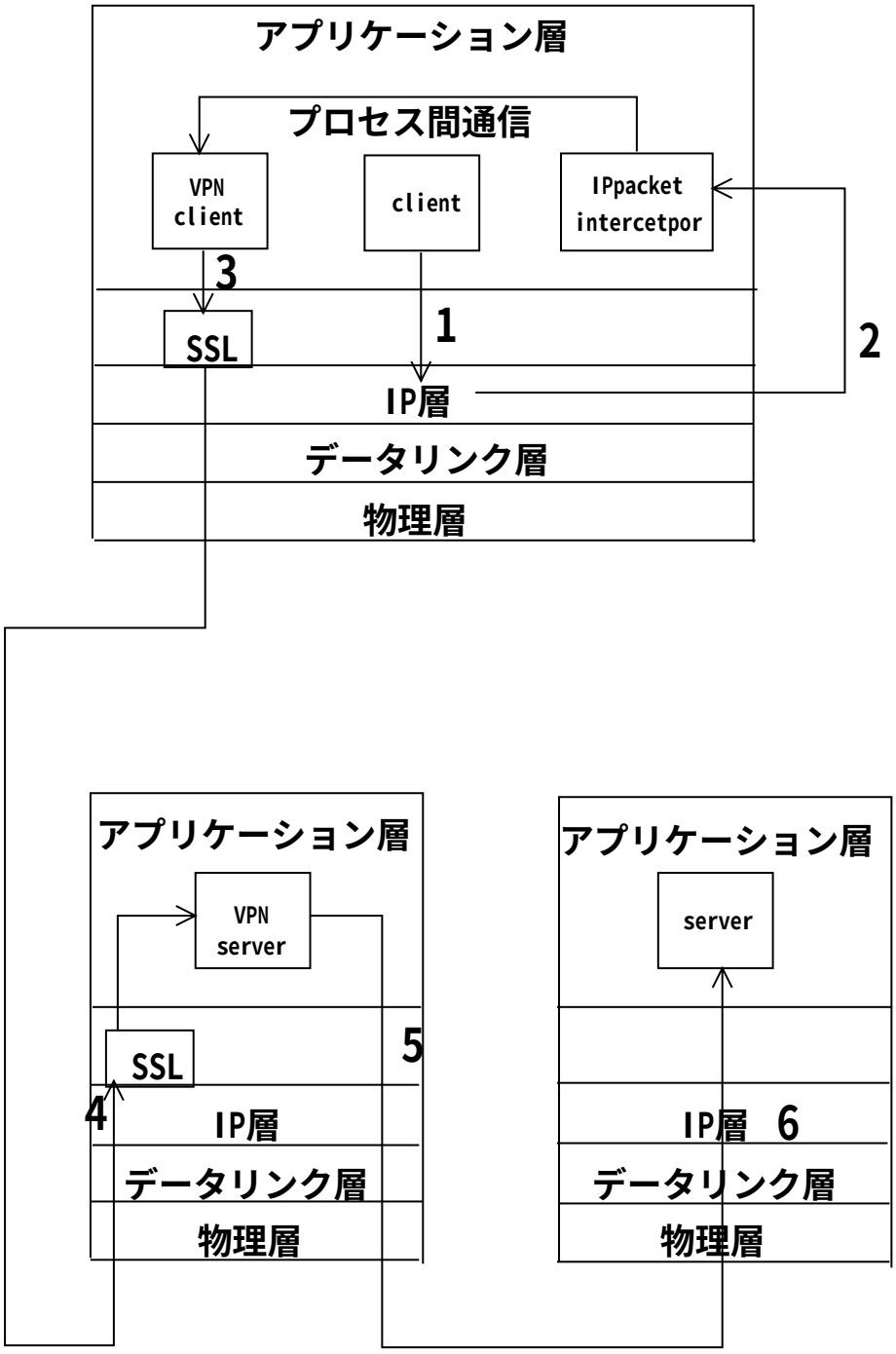

図 4.3: IP パケットがカーネル空間とユーザー空間との間を移動する回数（パーソナル VPN を利用した場合）

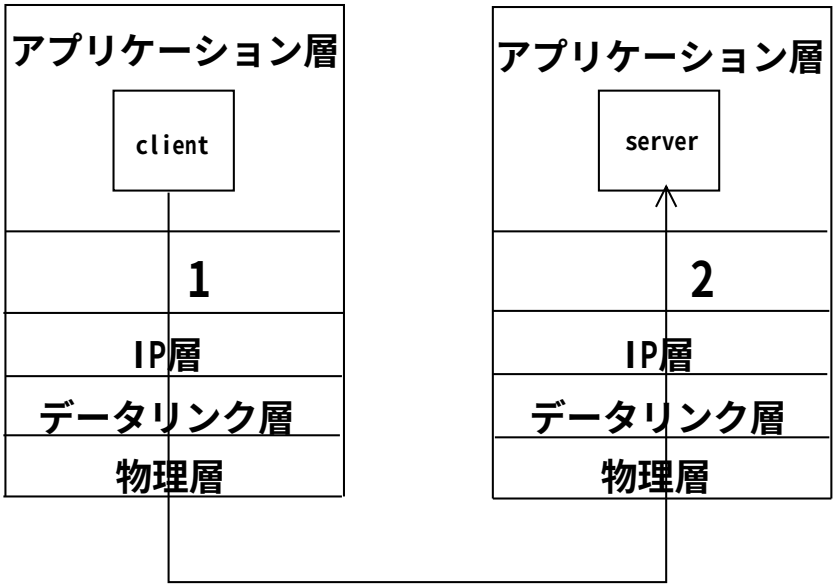

図 4.4: IP パケットがカーネル空間とユーザー空間との間を移動する回数（通所の場合）

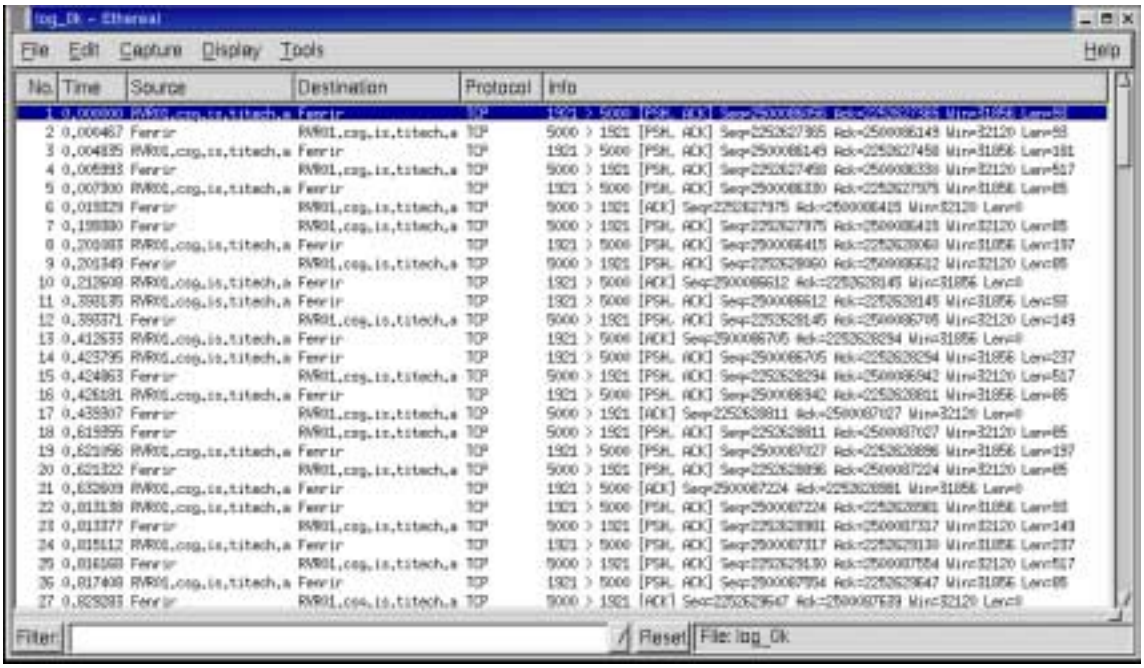

図 4.5: パケットの再送が発生する場合

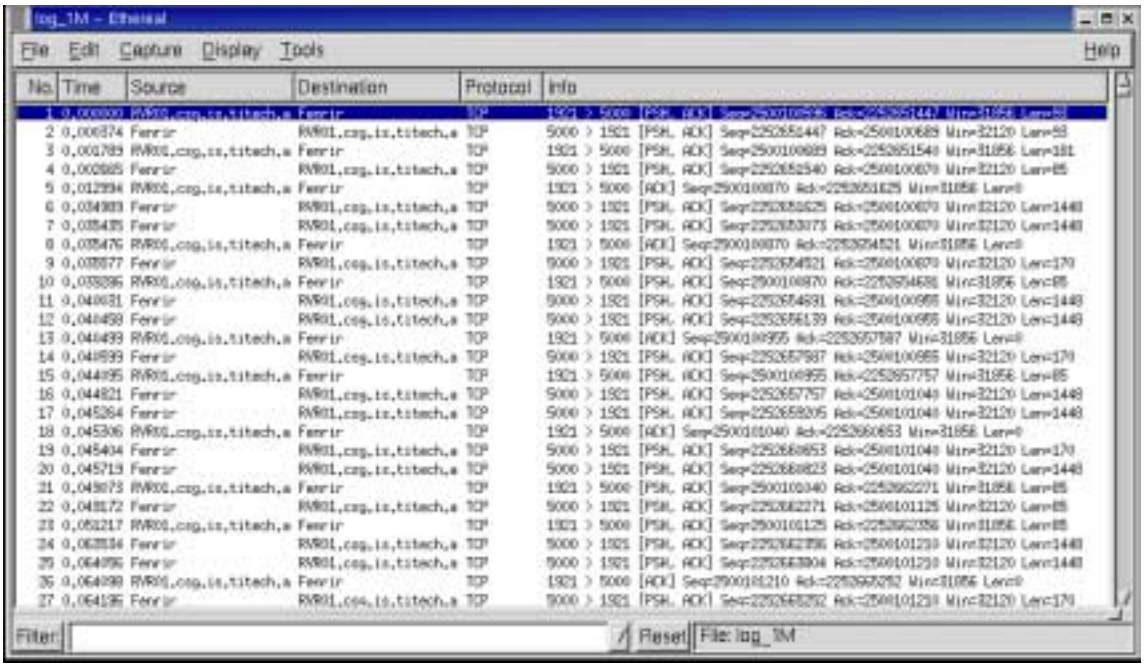

図 4.6: パケットの再送が発生しない場合

第5章　まとめ

本稿では仮想プライベートネットワークにユーザーの概念を取り入れたパーソナルVPNを提案した。パーソナルVPNは、IPパケットごとにユーザー認証を行うことで、特定のユーザーのみが利用可能にしたVPNである。我々はパーソナルVPNに必要な認証のうち、VPNの入り口でのユーザー認証の機能を実現した。また、直接通信ができないホスト間で通信ができるように、ゲートウェイとの間にパーソナルVPNを構築し、そこからrawソケットを使ってゲートウェイの先にあるネットワークと通信ができるようにした。

今後の課題を以下に列挙する

1. サーバー側でのユーザー認証の実装

   パーソナルVPNを構築するときに、VPNサーバーはその相手が誰であるか認証をする必要がある。OpenSSHのソースコードを参考にして、RSAユーザー認証を実装する予定である。

2. サーバーからの応答パケットの扱い

   パーソナルVPNを通ってきたIPパケットのカプセル化を解除したあと、そのIPパケットはrawソケットを使ってサーバーに送信する。今回は、サーバーからの応答パケットのIPアドレスとポート番号を利用して、パーソナルVPNを通過させるかどうか判断していたが、パーソナルVPNのユーザーが複数になると問題が生じる。

3. パーソナルVPN連結機能の実装

   今回実装したパーソナルVPNでは、直接通信ができないホスト同士の間にパーソナルVPNを構築することはできなかった。ゲートウェイまでパーソナルVPNを張り、ゲートウェイから先は、rawソケットを使いIPパケットを送る、という形態であった。そのため、

ゲートウェイから先は生の IP パケットが流れてしまう。パーソナル VPN を連結する機能があれば、直接通信できないホスト間にパーソナル VPN を構築できる。

4. 特定の条件において IP パケットの再送が頻繁に生じる問題の解決

   実験結果より、実装に何らかの問題があり、特定の条件で IP パケットの再送が頻発し、パフォーマンスが低下する問題があることがわかった。原因を探しバグを取り除く。

5. 実装をカーネル内部に行うことでオーバーヘッドを減らし、パフォーマンスを改善する

   divert ソケットを利用して IP パケットをインターセプトしているので、カーネル空間からユーザー空間へデータが移動するためオーバーヘッドが大きくなる。カーネルに手を加えないで実装できるという利点はあるが、パフォーマンスのことを考えると、カーネル内部で処理を行うようにしたほうがよいと考える。

6. 複数のユーザーが同時にパーソナル VPN を構築できるようにする

   今回の実装では、ユーザーごとに SSL セッションを作成し、ユーザーごとの VPN を作成することができなかったため、同時に複数のユーザーがパーソナル VPN を構築することはできなかった。ユーザーごとに SSL セッションを作成し、それぞれの SSL セッションとユーザー ID を対応させることにより、ユーザーごとに別々の SSL 暗号化通信路を作成できるようにする。

参考文献

[1] Atkinson, R.: Security Architecture for the Internet Protocol, Request For Comments(Standards Track)1825 (1995).

[2] Beck, M., Bohme, H., Dziadzka, M., Kunitz, U. and Verworner, D.: Linux カーネルインターナル, ピアソン・エデュケーション (1999).

[3] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W. and Zorn, G.: Point-to-Point Tunneling Protocol (PPTP), Request For Comments2637 (1999).

[4] Kent, S.: IP Authentication Header, Request For Comments(Standards Track)2402 (1998).

[5] Kent, S.: Security Architecture for the Internet Protocol, Request For Comments(Standards Track)2401 (1998).

[6] Kent, S. and Atkinson, R.: IP Encapsulating Security Payload (ESP), Request For Comments(Standards Track)2406 (1998).

[7] OpenSSH: `http://www.openssh.com/`.

[8] Project, T. O.: OpenSSL Project, `http://www.openssl.org/`.

[9] Townsley, W., Valencia, A., Rubens, J., Pall, G., Zorn, G., Palter, B. and Palter, B.: Layer Two Tunneling Protocol "L2TP", Request For Comments(Standards Track)2661 (1999).

[10] Valencia, A., Littlewood, M. and Kolar, T.: Cisco Layer Two Forwarding (Protocol) "L2F", Request For Comments2341 (1998).

[11] Verma, R., Verma, M. and Carlson, J.: L2TP Disconnect Cause Information, Request For Comments(Standards Track)3145 (2001).

[12] 光来健一, 千葉滋: インターネットにおけるパーソナルネットワークの構築, *SIG notes of Information Processing Sciety of Japan(2001-OS-88)*, pp. 83–90 (2001).

[13] 廣津登志夫, 福田健介, 明石修, 佐藤孝治, 山崎憲一, 菅原俊治: 仮想データリンクを用いた多重通信クラスに関する一考察, 第 3 回インターネットテクノロジーワークショップ (WIT2000) (2000).