卒業論文　　2004年度 (平成16年度)

ELA: 分散型仮想ネットワークの設計と実装

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　政樹

慶應義塾大学環境情報学部

青柳 禎矩

卒業論文要旨　　2004年度 (平成16年度)

ELA: 分散型仮想ネットワークの設計と実装

概要

情報共有や協調作業を目的として、多くの企業や学術機関においてLAN (Local Area Network) が構築されている。企業の支社のLAN同士で機密情報のやりとりを頻繁に行う場合などは、専用回線などの物理的回線を敷設し、LAN同士を接続する手法が用いられる。この手法は途中経路における情報漏洩の危険性がなくなるが、コストが大きいという問題がある。そこで近年は公衆通信網上に仮想の専用回線を構築可能なVPN (Virtual Private Network) が注目されている。VPNは通信経路としてインターネットなどの公衆通信網を利用するため、従来の手法と比べてコストが低いという特徴がある。

インターネットには途中経路における盗聴や改竄、成りすましなどのセキュリティリスクが存在するが、VPNは通信内容の暗号化および認証を行うため、専用回線に匹敵するセキュリティの高さを持つ。そのためLANに対して遠隔地から接続するという本来の目的だけでなく、ノード同士がインターネットを介した通信時の通信基盤としてVPNを利用することも有用である。しかしノードが多数存在する場合、従来のVPN機構では即興的なVPNは構築不可能である。

そこで本論文では、異種ネットワークセグメントに属するノード同士によって自律的に構築されたP2P (Peer-to-Peer) ネットワーク上でVPNを実現し、即興的に仮想的なLANを実現するELA (Everywhere Local Area network) を提案する。ELAを用いることによって多数のユーザノードの通信基盤としてVPNを即興的に構築でき、ユーザ間のセキュアな通信を実現する。

本機構をRed Hat Linux 9.0上でプロトタイプ実装し、複数の実機を用いた評価を行った。

慶應義塾大学　環境情報学部
青柳 禎矩

Abstract of Bachelor's Thesis
ELA: Design and Implementation of A Distributed Virtual Network

Abstract

Local Area Networks (LANs) are constructed at many places such as corporations or universities to support a sharing of knowledge and cooperative work smoothly. Cooperative LANs connect each other by a leased line to communicate confidential information between cooperative LANs. Although a leased line includes no risk of wiretap, it costs high. Thus, VPN (Virtual Private Network) which constructs a private network through a public network attracts attention.

The Internet includes risks of security such as wiretap, alteration, and spoofing. VPN is as secure as leased line because VPN system does encryption and authentication of data. As VPN uses a public network as communication pathway, it costs lower than other ways. So VPN is useful for not only to connect to remote LAN but also to use as a secure base of communication to communicate through the Internet. However, existing VPN systems can not construct VPN extemporarily if there are many nodes of users.

In this paper, we propose Everywhere Local Area network (ELA) which enables a fully distributed VPN system over Peer-to-Peer (P2P) network with nodes at different network segments. ELA enables to construct VPN for many users as a base of communication, and to communicate securely between nodes. I have implemented a proto-type of ELA on Red Hat Linux, and have evaluated it using physical nodes.

Sadanori Aoyagi

Faculty of Environmental Information
Keio University

目 次

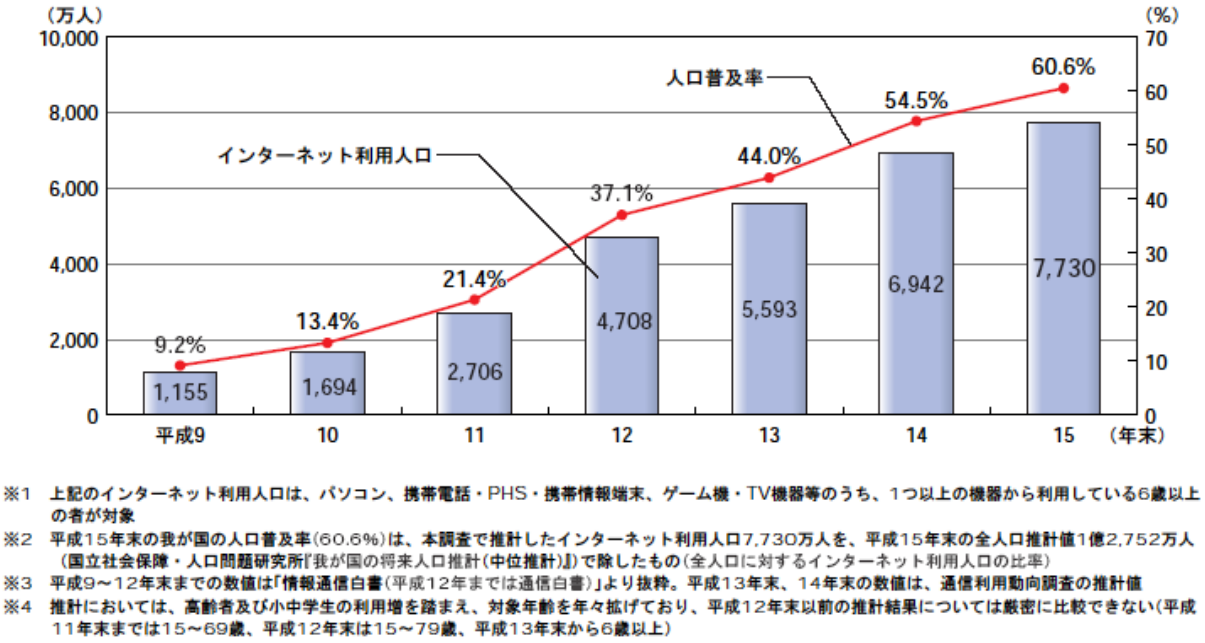
図 目 次

表 目 次

第1章　序論

1.1　動機

近年、常時広域接続のサービスの普及により日本におけるインターネット利用人口は増加している。図 1.1 が示すように平成 15 年度では 7730 万人がインターネットを利用し、国民 5 人のうち 3 人が利用していることになる。企業や研究機関のためだけではなく、インターネットは国民の生活に欠かすことの出来ない通信インフラストラクチャになった。



※1　上記のインターネット利用人口は、パソコン、携帯電話・PHS・携帯情報端末、ゲーム機・TV機器等のうち、1つ以上の機器から利用している6歳以上の者が対象
※2　平成15年末の我が国の人口普及率(60.6%)は、本調査で推計したインターネット利用人口7,730万人を、平成15年末の全人口推計値1億2,752万人（国立社会保障・人口問題研究所「我が国の将来人口推計(中位推計)」）で除したもの(全人口に対するインターネット利用人口の比率)
※3　平成9～12年末までの数値は「情報通信白書(平成12年までは通信白書)」より抜粋。平成13年末、14年末の数値は、通信利用動向調査の推計値
※4　推計においては、高齢者及び小中学生の利用増を踏まえ、対象年齢を年々拡げており、平成12年末以前の推計結果については厳密に比較できない(平成11年末までは15～69歳、平成12年末は15～79歳、平成13年末から6歳以上)

図 1.1: インターネット利用人口および人口普及率の推移（情報通信白書 平成 16 年度版より抜粋）

一般的なユーザのインターネットの主な利用目的は WWW (World Wide Web) による情報収集と他のユーザとのコミュニケーションの二つに分類できる。後者のコミュニケーション手段として、電子メールや WWW における掲示板システムが主流であるが、近年の端末の高性能化およびネットワークの広帯域化により新しいコミュニケーション手段が登場し始めている。例として IM (Instant Messenger) やビデオチャットなどが挙げられ、これらはユーザのノード同士が直接通信することによりリアルタイムなコミュニケーションを実現できる。今後更なる端末の高性能化およびネットワー

クの広帯域化により、これらのコミュニケーションツールはさらに普及していくものと考えられる。

しかし、インターネットを介した通信にはセキュリティリスクが存在する。例えば途中経路における通信の傍受により、パスワードなどの重要な情報が悪意ある人物に取得されることが例として挙げられる。図 1.2 が示すように、ユーザの 87.6% がコミュニケーションなどのサービスのセキュリティが重要であると考えている。セキュリティリスクを回避するには、コミュニケーションツールごとにセキュリティの設定をユーザが行う必要がある。しかしこの設定は手間がかかることが多く、図 1.2 が示すようにユーザの 86.2% がサービス利用時に手間をかけたくないと考えている。このように、ユーザはインターネットにおけるセキュリティリスクを回避したいが手間をかけたくないという相反した要求を持っている。

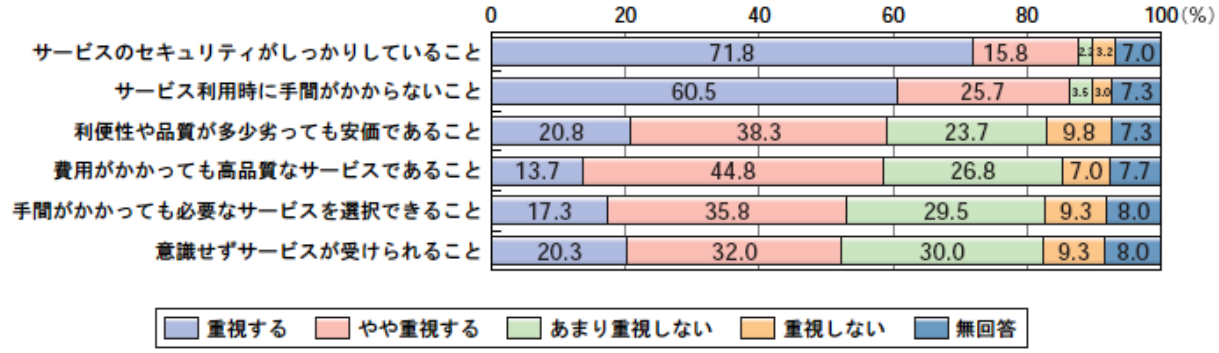


図 1.2: ネットワークを通じたサービスに対する考え方（情報通信白書 平成 16 年度版より抜粋）

そこで、インターネットにおけるセキュリティと手間の削減を両立する通信基盤を構築する、ということが本研究の動機である。ユーザ同士の全通信をセキュアにするには、セキュアな通信基盤を形成することが最適であると考えられる。なぜなら現在存在するコミュニケーションツールだけではなく将来登場する様々なアプリケーションにも対応できるためである。またその通信基盤はユーザが手間をかけずに即興的に構築できる必要がある。

このように、ユーザが透過的にセキュアな通信を実現可能な通信基盤によって、インターネットにおけるセキュリティの確保と手間の削減を両立できる。今後のさらなるアプリケーションの登場により、その通信基盤の重要性はますます高まると考えられる。

1.2 本研究の目的

本研究の目的は、ユーザ同士がインターネットを経由する通信を行う際のセキュアな通信基盤を即興的に構築することである。この通信基盤として本研究では VPN を用いる。ユーザが他のユーザと通信する場合は VPN を構築することにより、ユーザ

間のセキュアな通信を実現する。従来の VPN 機構では、多数の VPN が存在する場合は設定が面倒だったり、高性能かつ高信頼の端末を常時設置する必要があったりなど、多数のユーザ間に VPN を形成することが困難であった。

本研究では以上の目的を達成する機構である ELA (Everywhere Local Area network) を提案する。ELA は VPN の即興的な構築のために、ノードが多数存在する場合でも VPN を自動形成することにより、ユーザは意識せず VPN による通信が可能となる。また、ELA は単一ノードの故障により VPN 全体が利用不可にならないよう、規模性および耐故障性について考慮する。本論文では ELA の設計、実装および評価を行う。

1.3 本論文の構成

本論文は次のように構成される。本章である第 1 章では本研究の動機および目的について述べた。第 2 章では本研究の背景として LAN 同士の接続手法を説明し、その一手法である VPN の優位性について述べる。第 3 章では VPN の新しい利用手法を提案し、従来の VPN 機構では対応不可能であることを指摘する。そしてその解決策を提案し、その提案に類似する関連研究について述べる。第 4 章では分散型 VPN 機構である ELA の設計方針について述べ、第 5 章で ELA の実装手法を述べる。第 6 章においてプロトタイプ実装した ELA の評価を行い、オーバヘッドを計測する。第 7 章で本論文をまとめ、今後の課題について展望する。

第2章　背景

本章では本研究の背景を述べる。まず本研究の目的の通信基盤として構築するVPNについて概説する。次にそのVPNを構築する機構を分類し、その特徴を述べる。最後にVPNを構築する通信プロトコルの説明を行う。

2.1　VPNの概要

従来、企業の本社と支社のLAN間で通信する場合などでは、専用回線などの物理的回線を拠点間に敷設していた。LAN間通信にインターネットを経由しないため、途中経路における通信の盗聴や改竄などのセキュリティリスクがなく、プライベートIPアドレス同士の通信も可能だった。しかし、これらの物理的回線を用いた手法は敷設するコストがかかるという欠点がある。また回線を敷設した場所のみでしか利用できず、拠点の新設や移設の場合、それに伴い物理的回線の新設や移設の必要があり手間がかかる。

しかし、インターネットなどの公衆通信網上において仮想の専用通信経路を構築可能なVPNが登場した。従来利用されてきた専用回線やフレームリレーに比べ、VPNには以下のような利点がある。

- 低コスト

  専用回線などの手法で拠点間を接続する場合、拠点間に物理的な回線を敷設する必要があるため導入および維持に必要なコストが大きい。VPNは拠点間の通信回線としてインターネットを利用するため、通信回線の新規導入する必要がなくコストが小さい。

- 柔軟性

  専用回線で拠点間を接続する場合、拠点が移動する場合には専用回線を移設する必要がある。また拠点が新設する場合には専用回線も新設する必要がある。VPNはインターネットに接続可能な環境であれば利用可能で、拠点の移動や新設にも柔軟に対応できる。

VPNの需要は年々増大し続けている。2001年から2003年にかけての企業間通信網構築率を図2.1に示す。現在では半分以上の企業が企業間通信を利用していることが確認でき、利用は年々高まっていることが分かる。企業間通信網として低コストと柔軟性という特徴をもつVPNはこれからさらに普及していくことが予想できる。

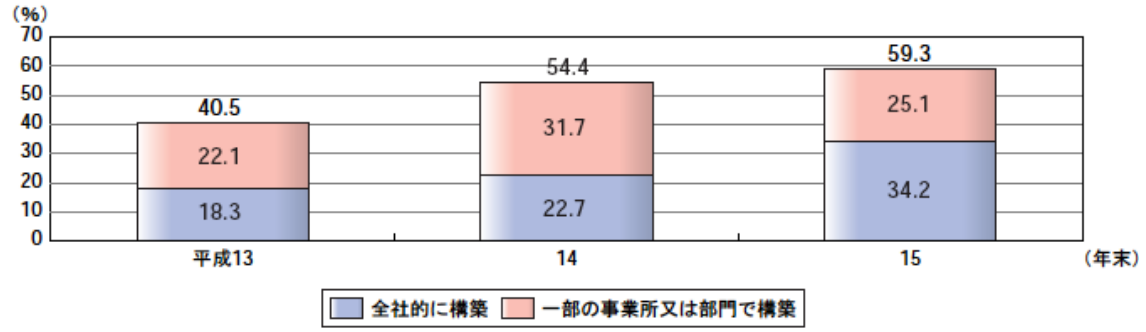


図 2.1: 企業間通信網の構築状況の推移 (情報通信白書 平成 15 年度版より抜粋)

2.2 VPN の動作概要

VPN では仮想性回線を構築するためにトンネリングという技術が用いられる。専用回線は拠点間を専用回線で接続することにより、拠点内のノード同士はインターネットなどの公衆通信網を経由せずに通信できる。それと同様に、VPN は拠点間にトンネルを形成して接続することにより、専用線と同様に拠点間の接続を実現する。図 2.2 が示すように、拠点間を VPN で接続したネットワークは拠点間を LAN ケーブルや専用回線などで接続したネットワークと論理的に等価である。

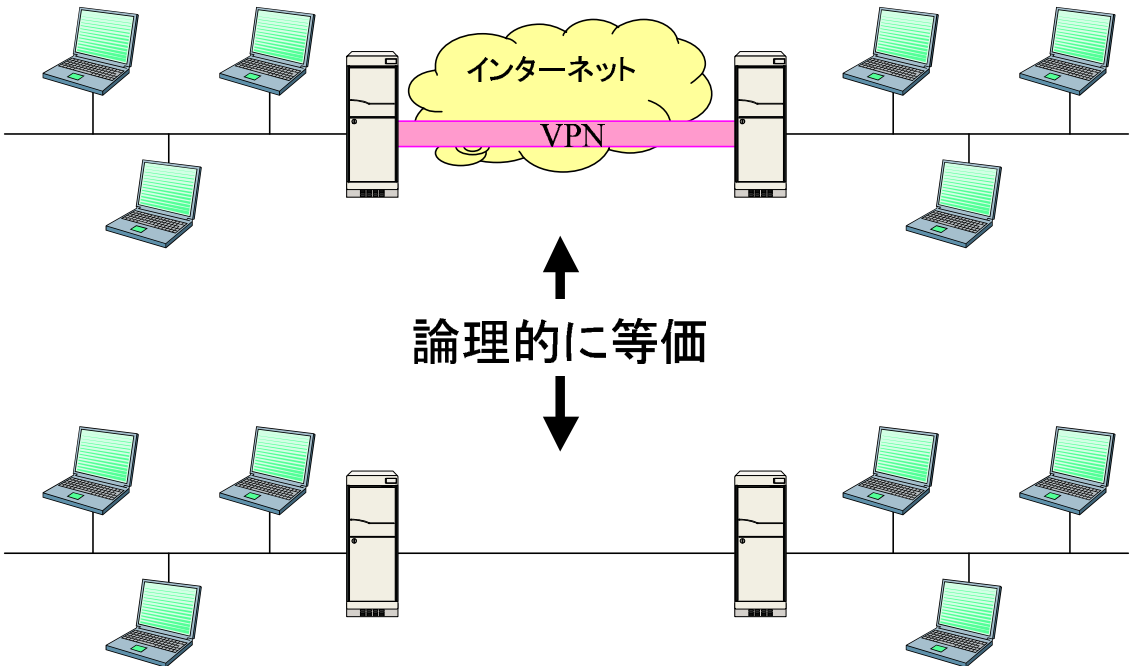


図 2.2: トンネルによって接続された拠点間

トンネルを形成し、通信内容を転送する通信プロトコルを本論文ではトンネリングプロトコルと定義する。トンネリングプロトコルは独自のヘッダを通信内容に付加し、

通信内容をカプセリングする。カプセリングされた通信内容は通信データとして扱われるため、トンネルを経由したプライベート IP アドレス宛の IP パケット送信も可能である。

専用回線と違い、インターネットは公衆通信網であるため世界中の人々が利用する。そのため途中経路における通信の傍受などのセキュリティリスクが存在する。VPN によって形成されたトンネルを専用回線と同等にセキュアにするために、以下のような処理が行われる。

- 暗号化

  途中経路において通信を傍受されたとしても、内容が判別されないように通信内容の暗号化を行う。

- 完全性の検証

  途中経路において通信を改竄された場合、改竄の検知するために完全性の検証を行う。

- 認証

  拠点に存在する拠点以外からの通信を防ぐため、通信発信元の認証を行う。

2.3 VPN 機構の分類

現在は数多くの VPN 機構が利用されている。どの VPN 機構も仮想の専用通信経路を提供する点で共通するが、VPN の利用目的や運用形態によって分類できる。

2.3.1 利用形態による分類

VPN を利用形態によって分類すると、VLL (Virtual Leased Lines), VPRN (Virtual Private Routed Network), VPDN (Virtual Private Dial-up Network) の 3 種類に分類でき、これらは RFC 2764 [4] において定義されている。それぞれの概要について簡単に述べる。

1. VLL (Virtual Leased Lines)

   VLL は拠点同士を 1 対 1 で接続する VPN である。従来の専用回線の代替として利用される。

2. VPRN (Virtual Private Routed Network)

   VPRN は拠点同士を多対多で相互接続でき、自身がルーティング機能をもつ VPN である。従来のフレームリレーの代替として利用される。

3. VPDN (Virtual Private Dial-up Network)

   VPDN は外出先や自宅などから遠隔地に存在する拠点へ一時的に接続する VPN である。従来のダイヤルアップの代替として利用される。VPDN によって拠点に接続することはリモートアクセスとも呼ばれる。

2.3.2 運用形態による分類

VPN を運用形態によって分類すると、Point-to-Point 型と Client/Server 型に分類できる。それぞれの概要と長所・短所について簡単に述べる。

- Point-to-Point 型

  拠点間を 1 対 1 で接続する VPN であり、VLL や VPDN の一部がこの運用形態に当てはまる。Point-to-Point 型 VPN は少数の拠点を接続する場合に利用される。例えば、図 2.3 (a) のように企業の本社と一つの支社間を接続するケース、図 2.3 (b) のように家庭のホームネットワークへリモートアクセスするケースなどが想定される。またルーティングを行わず処理が高速であることから、拠点間接続の基幹部分のみに利用されることもある。

  しかし、多数の拠点間をフルメッシュ型のトポロジで接続する場合、拠点数を N とすると構築する VPN の数は計算式 2.1 によって求まる。

$$X = \frac{N(N-1)}{2} \tag{2.1}$$

  例えば 100 の拠点同士をフルメッシュ型で接続すると、4950 もの VPN を構築する必要がある。拠点間接続する場合、VPN をフルメッシュに接続する必要はないが、部分的にメッシュを構築するだけでも多くの VPN を構築する必要がある。VPN 一つ一つに対して IP アドレスを設定し、ルーティングテーブルに追加する必要があるため手間が増大する。このため多数の拠点を相互接続する目的には利用しにくい。

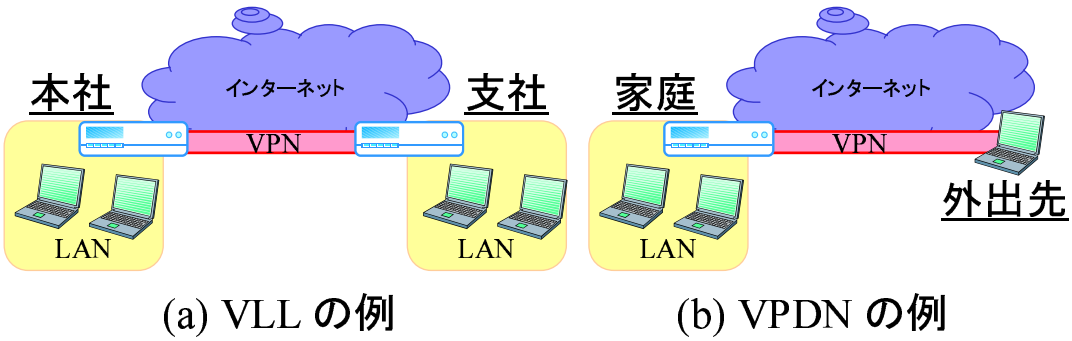


図 2.3: Point-to-Point 型 VPN 概要図

- Client/Server 型

  拠点間を 1 対多で接続する VPN であり、VPRN や VPDN の一部がこの運用形態に当てはまる。Client/Server 型 VPN は多数の拠点で接続する場合に利用される。例えば、図 2.4 (a) のように企業の本社と多数の支社間を接続するケース、図 2.4 (b) のように企業の LAN に多数の社員がリモートアクセスするケースなどが想定される。ルーティング処理は VPN 機構自身が備えているため、拠点が増加して VPN を新設する場合に手動でルーティング設定を行う必要がない。

  しかし Client 同士の通信を中継する Server が問題になる場合がある。Client 同士のトラフィックは全て Server を経由するため、Server の帯域や処理性能が十分でない場合は Server がボトルネックになる可能性がある。また Server は単一故障点であるため、もし Server が不慮の事故などにより動作しなくなると Client 間は全く通信できなくなる。このように Client/Server 型 VPN は Server に高性能と信頼性が要求されるため、Server の導入や維持に大きなコストや手間を必要とする欠点が存在する。

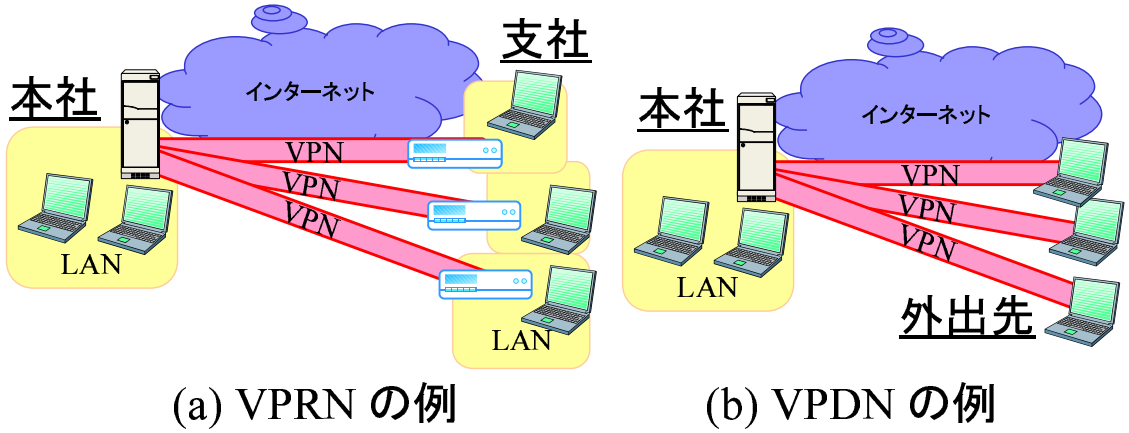


図 2.4: Client/Server 型 VPN 概要図

2.4 トンネリングプロトコル

前節までは VPN 機構の分類を行い、その説明を述べた。本節では VPN 機構がトンネルの形成に使用するトンネリングプロトコルについて説明する。VPN 機構の設計方針に応じて使用されるトンネリングプロトコルは異なる。VPN におけるトンネルの形成を行うトンネリングプロトコルと、そのトンネリングプロトコルを使用している VPN 機構を表 2.1 に示す。

- IPSec (Internet Protocol Security)

表 2.1: トンネリングプロトコルによる VPN 機構の分類

転送プロトコル	VPN 機構
IPSec	L2TP [20]
GRE	PPTP [13]
TCP	SoftEther [7], TinyVPN [21], Emotion Link [1], OpenVPN [22], VTun [14]
UDP	CIPE [17], OpenVPN [22], VTun [14]

IPSec [2] は IP レイヤにおける通信のセキュア化を目指し、IETF(Internet Engineering Task Force) において標準化が行われている。IPSec は認証や暗号、鍵交換などの複数のプロトコルを総称しており、認証アルゴリズムとして MD5 (Message Digest 5) や SHA-1 (Secure Hash Algorithm 1)、暗号アルゴリズムとして DES (Data Encryption Standard) や 3DES (Triple DES) など、鍵交換アルゴリズムとして IKE (Internet Key Exchange) を利用する。IPSec はノードの処理性能や求められるセキュリティ強度に応じて各アルゴリズムを変更できる。次世代のネットワーク層プロトコルとして注目を集める IPv6 において IPSec は必須の機能とされ、今後は普及が進むと予測される。

IPSec の欠点としては以下の 2 点があげられる。1 点目は TCP や UDP に比べ、IPSec をサポートする OS は多くないという点である。現段階では IPSec のプロトコルスタックが標準で搭載しない OS も数多くあり、そのような OS では IPSec の導入が必要である。2 点目は NAT (Network Address Translation) [12] 環境における利用が困難な点である。NAT とはプライベート IP アドレスしか割り当てられないノードからもインターネットに透過的にアクセス可能にする技術である。NAT はルータにおいて IP ヘッダの IP アドレスフィールドを、プライベート IP アドレスとグローバル IP アドレスに相互に改変することによって実現される。しかし IPSec は NAT による IP ヘッダの改変を改竄とみなしてしまうため、NAT 環境下のネットワークでは正常に動作しない。この問題を回避するため、IPSec パケットを UDP でカプセリングするという提案も存在するが、トンネリングに必要なヘッダがさらに増大するという新たな問題が発生する。

- GRE (Generic Routing Encapsulation)

  GRE [15] は汎用的なトンネリングプロトコルとして用いられる。GRE はトランスポート層のプロトコルで、GRE は暗号化などのセキュリティ機能がないため、VPN 機構による対応が必要である。

  GRE をトンネリングプロトコルとして用いる機構として PPTP (Point-to-Point Tunneling Protocol) [13] が挙げられる。PPTP はクライアント用 OS として幅広く利用されている Windows 2000 や Windows XP に標準でサポートされてい

る。PPTP は PPP (Point-to-Point Procotol) [19] フレームを GRE でカプセリングすることにより、トンネリングを可能にする。カプセリングされた PPP フレームは MPPE (Microsoft Point-to-Point Encryption) [8] によって暗号化できる。GRE の欠点として、NAT や Firewall が GRE に未対応の場合は利用できない点が挙げられる。

- TCP (Transmission Control Protocol)

  TCP [11] は VPN に限らず、HTTP や FTP など幅広いアプリケーションにおいて一般的に利用されているトランスポート層プロトコルである。TCP をトンネリングプロトコルとして用いる機構として、SoftEther [7], TinyVPN [21], Emotion Link [1], OpenVPN [22], VTun [14] が挙げられる。

  TCP をトンネリングプロトコルとして利用する長所として、TCP はコネクション型のプロトコルであることが挙げられる。Firewall や NAT 環境下に存在するノードでも外部のノードに対して TCP セッションを確立すれば、それ以降は双方向に通信可能である。逆に TCP を用いる欠点として、TCP over TCP [18] と呼ばれる問題の存在が挙げられる。TCP over TCP 問題とはアプリケーションが送信した TCP セグメントを含む通信内容を、VPN 機構が TCP を用いて転送する場合に発生し、途中経路における通信の紛失が発生すると極端に通信速度が低下する問題である。これは TCP による輻輳制御が通信内容に対して 2 度行われることが原因である。

- UDP (User Datagram Protocol)

  UDP [10] をトンネリングプロトコルとして用いる機構として、CIPE [17], OpenVPN [22], VTun [14] が挙げられる。TCP と同様に一般的なネットワークアプリケーションで用いられるトランスポート層のプロトコルである。UDP は TCP のようなフロー制御や輻輳制御などをしない。そのため TCP と比べ、パケット到達の保証がなく信頼性が低いが、処理が簡易であるため転送速度が高いという特徴がある。

このようにトンネリングプロトコルはそれぞれ特長がある。そのなかで、TCP と UDP は他の一般的なネットワークアプリケーションで使用される汎用的なトランスポート層のプロトコルである。最近の Windows や Linux などの OS はネットワークアプリケーションの使用を前提に開発されており、TCP と UDP のプロトコルスタックが標準で存在する。そのため転送プロトコルに TCP や UDP を用いる VPN 機構を導入する場合、ユーザがノードやルータに新規のプロトコルスタックの導入する必要がない。これは VPN 機構の導入という観点から考えて、大きな特長といえる。

通信制約のある環境における利用という観点から TCP と UDP を比較した場合、TCP はコネクション型プロトコルであるため、一度コネクションを構築すれば Firewall や NAT 環境においてもデータの送受信が可能である。UDP はコネクションレス型プロ

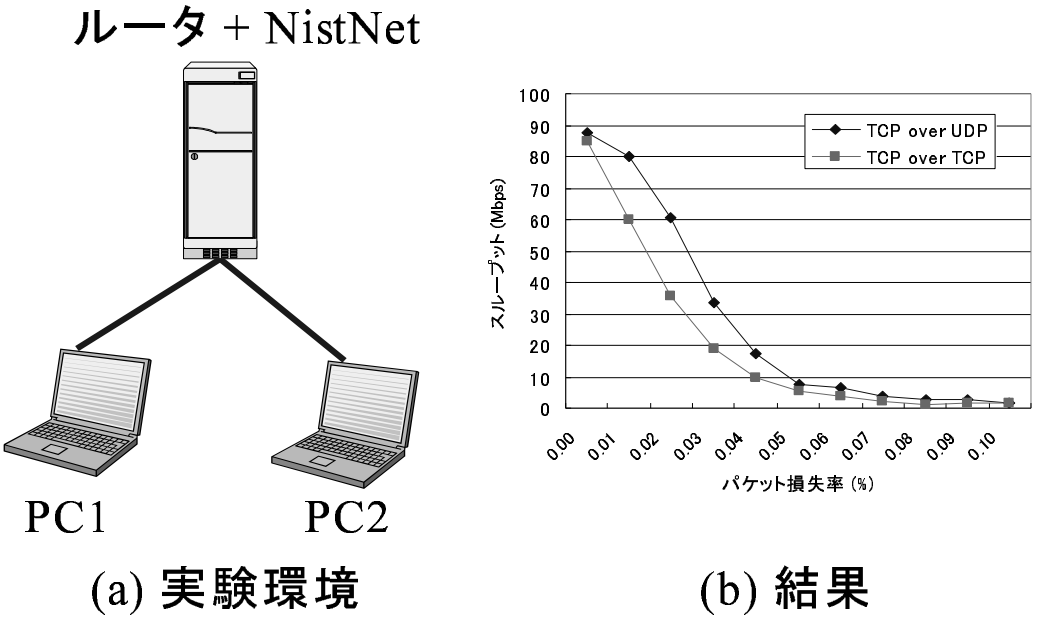


図 2.5: OpenVPN によるスループットの測定

トコルであるため、Firewall や NAT 環境においては UDP データグラムを送信可能だが受信不可能という場合がある。

性能という観点から TCP と UDP を比較した場合、UDP は簡素なプロトコルであるため処理が高速である。TCP は再送処理や輻輳制御などの処理が多く、また TCP over TCP の問題によって利用可能帯域が減少する。TCP と UDP の性能の差を検証するため、VPN 機構実装の一つである OpenVPN [22] を用いてトンネリングプロトコルの違いによるスループットの違いを検証した。OpenVPN はトンネリングプロトコルに TCP と UDP を明示的に選択して利用できる。

検証のため図 2.5(a) のような環境を用意した。ルータ (CPU: Celeron 433MHz, Memory 192MBytes, OS: Red Hat Linux 9) に NISTNet [6] を導入し、ルータで任意の割合でパケットを破棄できるようにした。ルータに 2 台の PC (CPU: Pentium M 1.7GHz, Memory 1GByte, OS: Red Hat Linux 9) を 100Base-TX で接続し、それぞれ異なるネットワークセグメントを割り当て、2 台の PC 間で OpenVPN を用いて VPN を構築した。そして Netperf を利用して TCP セグメントを発生させ、VPN 経由における 2 台の PC 間スループットを計測した。

実験結果を図 2.5(b) に示す。ほとんどのパケット損失率において、トンネリングプロトコルに UDP を利用した方がスループットが大きいという結果になった。パケット損失率が 0%の場合には両者のスループットは大きく変わらない。しかしルータにおいてパケットを損失させると TCP の場合はスループットが急激に減少した。特にパケット損失率が 1, 2%の時はスループットの差が 20Mbps 以上になった。パケット損失が起きると TCP は再送処理や輻輳制御を行うため、スループットが減少するものと考えられる。

第3章　問題意識

本章ではまずインターネットを介したユーザ間の通信基盤として VPN を利用することを提案する。次に既存の VPN 機構ではそのような VPN の構築が困難であることを示し、その解決策を説明する。最後にその関連研究について述べる。

3.1　通信基盤としての VPN 利用

現在多くの VPN 機構が提案され、一部の VPN 機構は実際に拠点間接続に利用されている。その目的は企業や学術機関などの LAN に存在するファイルサーバやメールサーバなどの既存のネットワークリソースを遠隔地から活用することである。しかし VPN によって構築される通信経路はセキュリティの面において優れており、その他の目的に利用することも可能である。

そこで VPN によって異種ネットワークセグメントに所属するユーザのノード間の通信基盤として VPN の利用も有用だと考えられる。従来の VPN のような既存の LAN に存在するネットワークリソースの利用が目的ではなく、インターネットを介したノード同士の直接通信のセキュリティをより強化することが目的である。そのためユーザが他のユーザと通信したいと思ったときに即興的に VPN を構築できる必要がある。

しかし現在提案されている VPN 機構はユーザのノード間で即興的に VPN を構築するのは困難である。その理由を以下に示す。

Point-to-Point 型 VPN 機構における問題

Point-to-Point 型 VPN 機構はユーザのノードを 1 対 1 で接続する。そのためノードが多数の場合は多くの VPN を構築する必要がある。もし N 台のノードをフルメッシュ型トポロジで VPN 構築する場合、$N(N-1)/2$ の VPN を構築する必要がある。また VPN セッションごとにルーティングを設定する必要がある。このように多数のユーザが存在する場合は VPN に必要な設定が多く、VPN を即興的に構築することはできない。

Client/Server 型 VPN 機構における問題

Client/Server 型 VPN 機構はユーザのノード間通信を中継するサーバが存在する。そのためノード同士が通信する場合は必ずサーバを経由する。多数ユーザのノード間で

VPN を構築する場合は以下に示す二つの問題が存在する。

一つ目は運用の問題である。VPN の構築には通信を中継するサーバが必要になるため、事前にサーバの設置が必要になる。サーバの導入および運用にはコストだけではなく手間もかかるため、VPN が必要になったときに即興的に構築できるとは言えない。もしユーザのノードの一つをサーバにして VPN を構築する場合は、どのノードをサーバにするかを決定しなければならない。

二つ目は性能の問題である。VPN における通信は必ずサーバを経由するため、サーバが利用可能な帯域以上の帯域をクライアントは利用できない。そのためサーバが通信のボトルネックとなる可能性がある。クライアントから他のクライアントへ巨大なファイルを送信することや、サーバがストリーミングビデオを再生することなどにより、VPN における通信速度が極端に遅くなるという現象が想定される。

VPN 機構全体における問題

2.4 節で述べたように、各トンネリングプロトコルには異なる特徴がある。特に TCP と UDP は Web ブラウザなどの一般的なネットワークアプリケーションで利用されており、既存のネットワークとの親和性が高い。最近の Windows や Linux などの OS には TCP と UDP のプロトコルスタックが標準で存在するため、VPN 機構の導入時にユーザが OS に新規のプロトコルスタックの導入が不要であることが挙げられる。

TCP は NAT や Firewall などの通信制約下においても、一度外部のノードに対して TCP セッションを確立すればデータの送受信が可能という長所がある。しかし、TCP は TCP over TCP による通信速度の低下という短所がある。UDP は簡素なプロトコルであるため TCP に比べ通信速度が高速という長所がある。しかし、UDP は Firewall や NAT などの通信制約により、外部のノードから通信を受信できない場合がある。そこで通信制約のない環境のノードは UDP による通信、通信制約によって UDP による通信ができないノードは TCP による通信を行うというように、通信制約に応じた最適なトンネリングプロトコルを利用できると効率が良い。

しかし、従来の VPN 機構における転送プロトコルは固定である。OpenVPN や VTun のようにユーザが転送プロトコルを事前に選択できる VPN 機構も存在する。2.4 節で述べたように転送プロトコルとして TCP と UDP を用いる場合、UDP を用いた方がスループットが大きい。そのため NAT などの通信制約が存在せず UDP を利用できるのであれば、転送プロトコルとして UDP を用いた方が好ましい。しかし VPN 機構が Firewall や NAT などの通信制約を考慮し、最適な転送プロトコルを自動的に選択する VPN 機構は存在しない。

3.2 解決策の提案

前節で述べたように、既存の VPN 機構は多数ユーザのノード間で VPN を即興的に構築できない。Clinet/Server 型 VPN 機構は Server の存在による性能と運用の問題が

ある。Point-to-Point 型 VPN 機構は Server は必要ないが、ユーザ自身で VPN のトポロジを形成する必要がある。これらが VPN の即興的構築の阻害要因となっている。

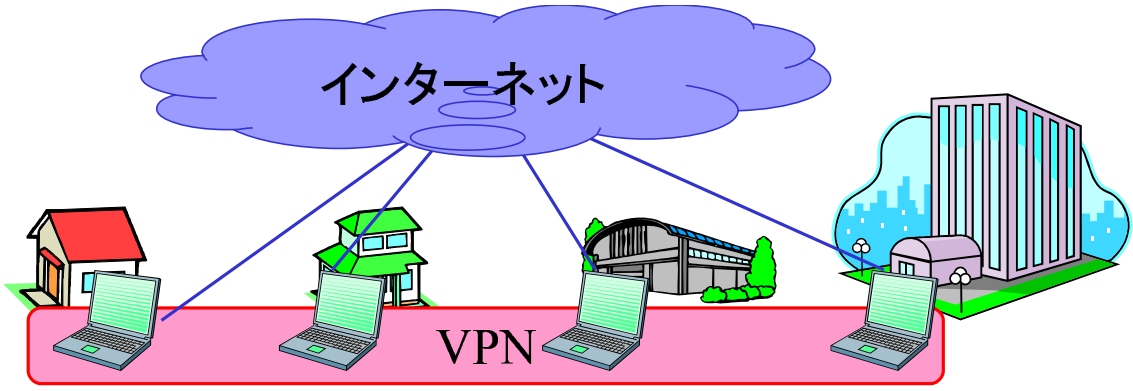


図 3.1: ユーザノード間の通信基盤としての VPN

「単一ノードに通信が集中しないトポロジ」を「自動形成」する VPN 機構が必要である。「単一ノードに通信が集中しないトポロジ」として近年 P2P 型が注目されている。P2P ネットワークを即興的に構築するため、VPN 機構が「自動形成」する必要がある。

そこで本論文では、P2P ネットワークトポロジの VPN 機構を各ノードが自律的に形成することにより VPN を実現する機構である ELA (Everywhere Local Area network) を提案する。ELA はインターネットを介したユーザ間の通信基盤としての利用を目的とした VPN 機構である。ELA のネットワークトポロジは単一ノードに通信が集中しない P2P 型であるため、単一ノードの故障により全てのノードが VPN による通信が不可能にならない。また、その P2P ネットワークは ELA によって自動形成される。

3.3 関連研究

ノード間通信の VPN を即興的に構築する機構として IVGMP (Internet VPN Group Management Protocol) [3] が挙げられる。ユーザの通信要求に応じて、VNOC (Virtual Network Operation Center) に IVGMP を用いて接続先および通信ポリシーを照会し、拠点間で IPsec による VPN を形成する。この機構はユーザが多数存在する場合でも VPN を自動形成し、かつ通信が単一ノードに集中しない。しかし VPN 通信開始時に必ず VNOC への照会が必要であるため、VNOC が単一故障点となる。

また、本論文では P2P トポロジによるオーバレイネットワーク上に VPN を構築することを目的としているが、オーバレイネットワーク上に VPN を構築する機構として [7], [21] が挙げられる。[7], [21] はスイッチングハブの役割をする Server に仮想的な NIC を持つ Client が接続しすることにより VPN を構築する。Server を中心としてオーバレイネットワークを形成することにより、多数のユーザ間で VPN を経由した通信が可能である。[7] は SSL、[21] は AES による通信の暗号化を行い、セキュリティを

確保する。これらの機構は Client 間の通信に必ず Server を必要とするため、前節で述べた Server における性能と運用の問題が発生する。

第4章　ELAの設計

本章ではELA (Everywhre Local Area network) について述べる。最初にELAの概要を述べ、次にELAが形成するネットワークを説明する。最後にELAのソフトウェアのモジュール構成を述べる。

4.1　ELAの概要

本節ではELA (Everywhere Local Area network) を提案する。ELAは異種ネットワークセグメントに属するノード同士がP2Pネットワークを形成し、即興的に仮想的なLANを構築する。その仮想的なLANはノード同士のセキュアな通信基盤として利用可能で、本論文ではELA-VPNと定義する。ELA-VPNのイメージを図4.1に示す。TCPおよびUDPによる通信の送受信可能なノードをコアノードとし、コアノード以外のNATやFirewallなどによって通信に制限があるノードをエッジノードとする。コアノードがメッシュネットワークを形成し、エッジノードがそれらのコアノードに接続するという規則に従い、P2Pネットワークトポロジを自動的に形成する。

ELAによって構築されるVPNは既存のネットワークセグメントから独立した仮想的なプライベートネットワークである。他のノードによってELA-VPNにおけるプライベートIPアドレスが各ノードに割り当てられる。

4.1.1　想定シナリオ

ELAを利用する想定シナリオとして以下があげられる。どちらのシナリオも利用ノードは30台くらいまでを想定する。

ユーザ間通信の透過的な暗号化

近年はサーバを介さずに、ユーザのノード同士が直接通信することによってコミュニケーションを実現するアプリケーションが増加しており、例としてIM (Instant Messenger) やビデオチャット、一部のネットワークゲームなどが挙げられる。これらのアプリケーションは通信経路の暗号化などを正しく設定しないと、途中経路で盗聴される危険性がある。不用意にパスワードなどの大事な情報を送信すると、悪意ある人物がそれらを盗聴し、悪用する可能性がある。

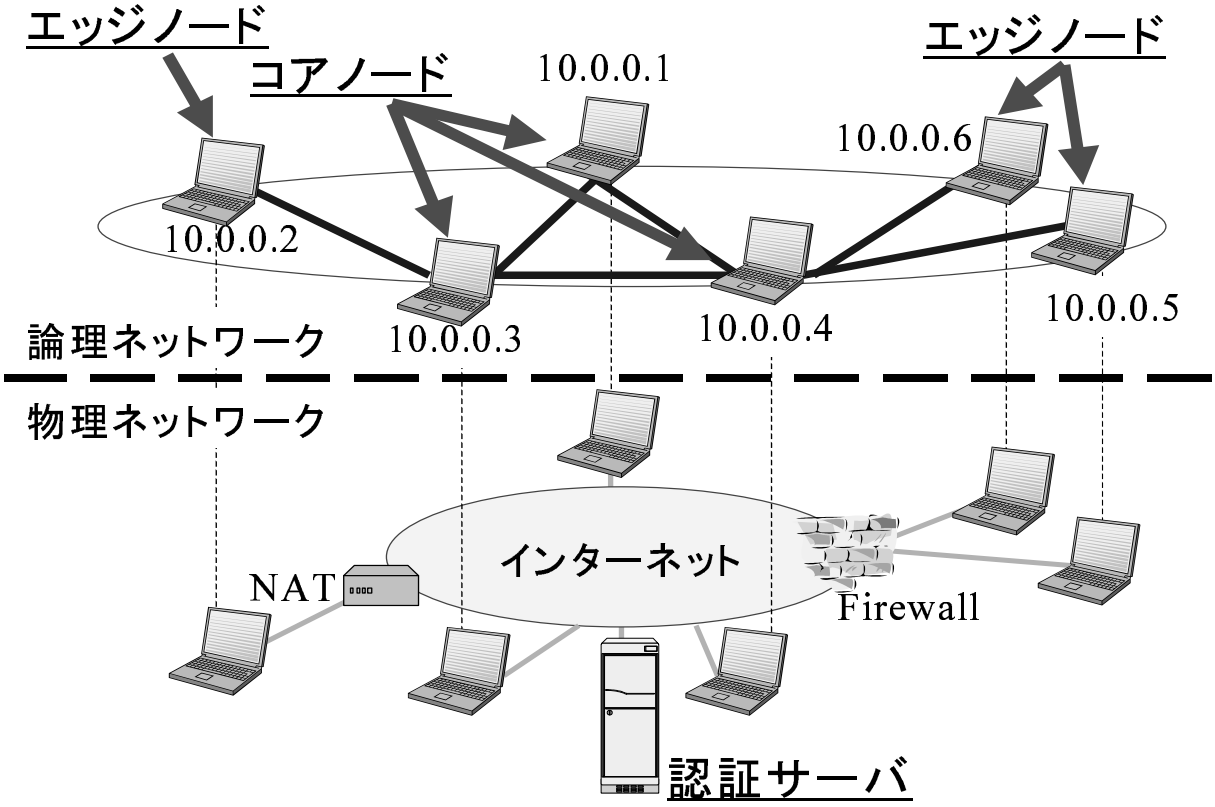


図 4.1: ELA によって構成される ELA-VPN のイメージ

そこで ELA を利用してユーザのノード間で即興的に VPN を構築し、各アプリケーションによる通信を透過的に暗号化する。

LAN 用アプリケーションの利用

閉塞的なネットワークである LAN での使用を前提としているアプリケーションがが存在する。Windows Network や Doom [9] などの一部のネットワークゲームは、通信相手のノードの探索にブロードキャストを用いる。そのためこれらのアプリケーションは互いのネットワークセグメントが異なると使用できず、インターネットを介した利用は不可能である。

NFS (Network File System) は同一ネットワークセグメントに所属するノードから利用され、通信経路は盗聴の危険がなく信頼可能という前提で設計が行われている。そのため通信経路における暗号化などが行われず、インターネット経由の利用はセキュリティの面から利用できない。

そこで ELA を利用してユーザのノード間で即興的に VPN を構築し、LAN での利用を目的としたアプリケーションをインターネット経由で利用可能にする。

4.1.2 機能用件

ELA はVPN 機構の一種であるが、P2P ネットワークトポロジを形成し、通信を単一ノードに集中させずに分散させるという特徴がある。VPN に必要な機能として悪意ある人物による成りすまし防止を行うユーザ認証機能、通信を暗号化し必要なヘッダを付加するカプセリング機能、カプセリングされた通信の経路を構築するトンネリング機能が挙げられる。また ELA の特徴である分散型を実現するため、P2P ネットワークトポロジを形成するトポロジ構築機能、ノードの参加および離脱などの情報をノード間で共有するためのメッセージ機能、P2P ネットワークにおいて通信の転送先を決定するルーティング機能が挙げられる。

これらを踏まえると、ELA に必要な機能は以下の項目になる。

- ユーザ認証機能
- カプセリング機能
- トンネリング機能
- トポロジ構築機能
- メッセージ機能
- ルーティング機能

次節では以上の各機能を説明する前に、トポロジ構築機能によって形成される P2P ネットワークを説明する。その後に、ELA の P2P ネットワーク形成に必要な各機能を述べる。

4.2 動作手順

本小節では P2P ネットワーク形成から VPN による通信に至るまでの、ELA の動作手順について説明する。

4.2.1 ノードの参加準備

1. ノードの発見

   まず既に ELA-VPN に参加しているノードを発見する。ノードの発見機能は ELA に組み込まれていない。ユーザは WWW やメールなどを利用してすでに ELA-VPN を構築しているユーザを発見し、インターネットにおけるノードの IP アドレスを訊ねる。まだ ELA-VPN を構築しているユーザがいなければ自身だけの ELA-VPN を構築し、他のノードから接続される準備を行う。**2.** を飛ばして **3.** にいく。

2. ユーザ認証

   ELA-VPN にノードが参加する場合、ELA-VPN に既に参加しているノードからユーザ認証をうける。ELA-VPN の参加ノードは参加が認可されたユーザのリストを共有しており、そのリストを用いて認証を行う。

3. プライベート IP アドレスの割り当て

   ELA-VPN におけるプライベート IP アドレスを割り当てる。新規参加するノードが特定のプライベート IP アドレスを利用したい場合、ELA-VPN の既存のノードと IP アドレスが重複しなければその IP アドレスを使用できる。重複があった場合、あるいは利用したい IP アドレスがない場合、利用されていないプライベート IP アドレスの中からランダムに決定する。

4. ノードの分類

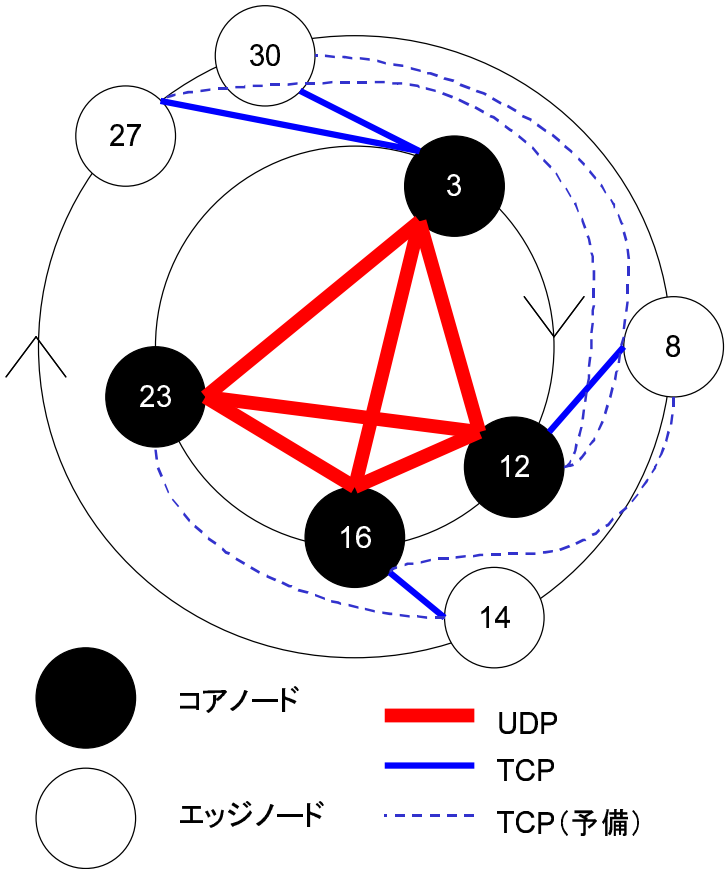
   各ノードはコアノードとエッジノードに分類される。コアノードは他のノードから TCP セッションを確立可能で、UDP によるデータグラムの送受信が可能である。エッジノードはコアノード以外のノードである。ノードの分類は ELA の P2P ネットワーク形成に関わり、コアノードかエッジノードかによって ELA-VPN における役割が異なる。

4.2.2 P2P ネットワークの形成

ELA-VPN の P2P ネットワークの例を図 4.2 に示す。ELA-VPN におけるプライベート IP アドレスのハッシュ値を ID とし、P2P ネットワークでは ID の小さい順に時計回りに配置される。P2P ネットワークの論理的に中央の位置にコアノードが配置され、その周りをエッジノードが囲む。

コアノード同士は互いに UDP で通信を行い、エッジノードはコアノードに TCP セッションを確立して通信を行う。各エッジノードが TCP セッションを確立する相手のコアノードを、そのエッジノードの「親」と呼ぶ。反対に各コアノードに TCP セッションを確立しているエッジノードを、そのコアノードの「子」と呼ぶ。例えば ID 30 の親は ID 3 のコアノード、ID 3 の子は ID 27 と ID 30 のエッジノードとなる。エッジノードが TCP セッションを確立するコアノードは、そのエッジノードの時計回りに ID が一番近いコアノードである。

コアノードが ELA-VPN から離脱した場合、そのコアノードの子も ELA-VPN から離脱してしまう。それを防ぐため、各エッジノードは親の次に ID が時計回りに近いコアノードに予備の TCP セッションを確立しておく。エッジノードが親のコアノードと TCP セッションを通じて通信できなくなった場合は、親のコアノードが ELA-VPN から離脱したと判断する。そして、今まで予備の TCP セッションを確立していたコアノードが親となり、エッジノードは新しく他のコアノードに対して予備の TCP セッションを確立する。コアノードが正規の終了手続きを経て ELA-VPN を離脱する場合は、



ルーティングテーブル

Key	Value
3	3のIPアドレス
8	12のIPアドレス
12	12のIPアドレス
14	16のIPアドレス
16	16のIPアドレス
23	23のIPアドレス
27	3のIPアドレス
30	3のIPアドレス

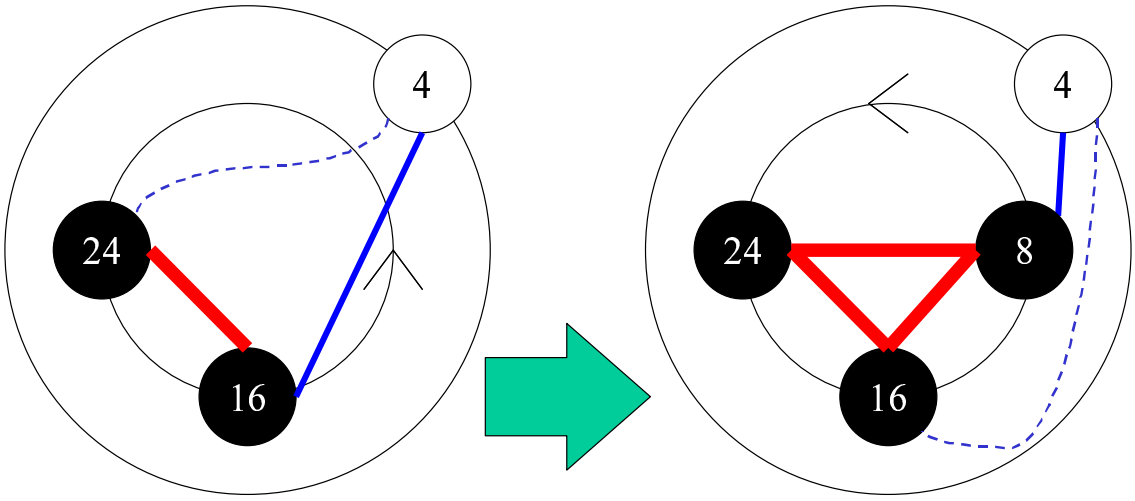
図 4.2: ELA の P2P ネットワーク

TCP セッションを明示的に切断することによりエッジノードも即座に親のコアノードを切り替えることができる。コアノードがネットワークトラブルにより突然 ELA-VPN を離脱する場合は、TCP セッションが明示的に切断されないため、エッジノードは通信のタイムアウト等から親が離脱したと判断する。

4.2.3 データ転送

ELA の P2P ネットワークはフルメッシュ型ではないため、論理的に直接接続していないノード同士は他のノードを経由して通信する必要がある。ELA では経由するノードを決定するルーティング処理が必要である。

ルーティング機構によって決定される途中経路と中継回数は発信元ノードと宛先ノードによって異なる。コアノード同士は UDP を用いて直接通信し (例：3 ⇔ 16)、コアノードと子のエッジノードは TCP セッションを用いて直接通信する (例：16 ⇔ 14)。これら以外の場合は、上記のパターンを組み合わせる。例えばコアノードと子ではないエッジノードと通信する場合、他のコアノードを経由して通信する (例：3 ⇔ 16 ⇔ 14)。エッジノード同士が通信する場合、コアノードを経由して通信する (例：27 ⇔ 3



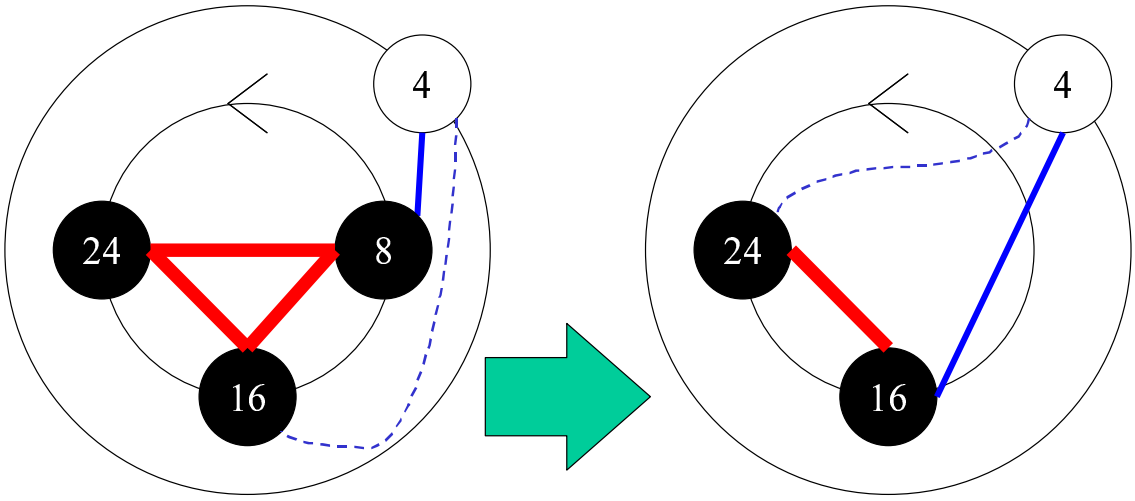
Key	Value
4	16のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

Key	Value
4	16のIPアドレス
8	8のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

図 4.3: コアノード参加時

⇔ 16 ⇔ 14)。

コアノードは他のコアノードか子のエッジノードに通信内容を転送するので、コアノードは通信内容の宛先によって転送先のノードを決定する必要がある。そのため各コアノードは二つのテーブルを保持する。一つは ELA-VPN に参加する全てのノードに関するルーティングテーブルで、*key* には ELA-VPN における IP アドレスのハッシュ値、*value* にはコアノードの場合はそのコアノードのインターネットにおける IP アドレス、エッジノードの場合は親のコアノードのインターネットにおける IP アドレスが格納される。このテーブルはルーティングテーブルを用いて、全コアノードで共有する。もう一つは各コアノードの子に関するテーブルで *key* には ELA-VPN における IP アドレスのハッシュ値、*value* には TCP セッションの識別子が格納されている。コアノードは他のノードからデータを転送された場合、これらのテーブルを参照して他のノードへ転送する必要がある場合はそのノードへ転送し、自分宛であれば受信する。エッジノードは P2P ネットワークの論理的に外側に位置するため、ルーティングに関して考慮する必要がない。



Key	Value
4	16のIPアドレス
8	8のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

Key	Value
4	16のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

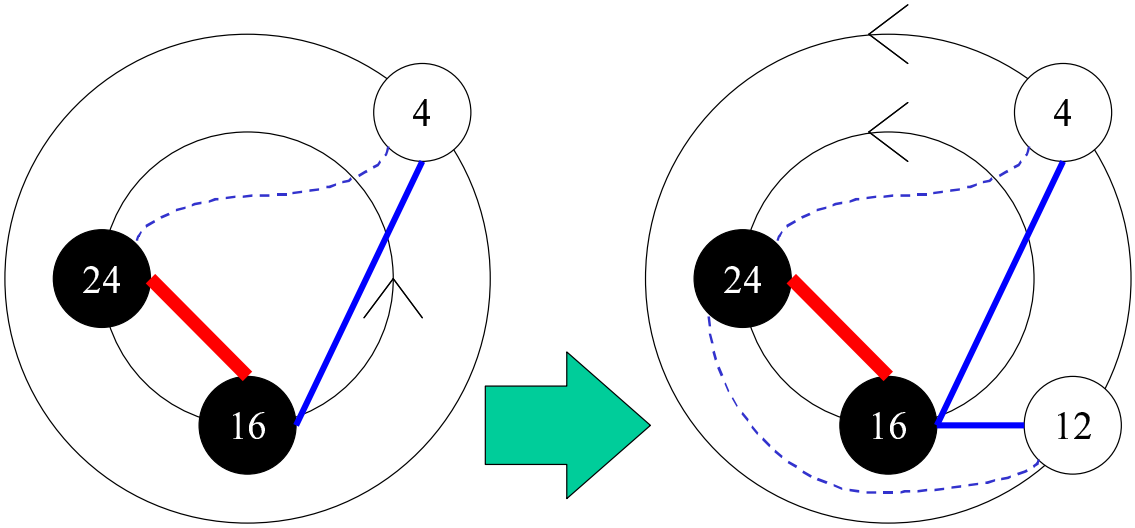
図 4.4: コアノード離脱時

4.2.4 P2P ネットワークの維持

各ノードはユーザの都合に応じて ELA-VPN へ参加および離脱する。その度にルーティングテーブルを修正し、P2P ネットワークにおいて正しくデータ転送できよう維持する。

コアノードが参加する例を図 4.3 を用いて示す。ELA-VPN に新規に ID 8 のコアノードが参加するものとする。ID 8 のコアノードは他のコアノードからルーティングテーブルを取得し、自分自身の情報を追加する。ID 8 のコアノードはルーティングテーブルを参照して、本来自分に対して TCP セッションを確立すべき ID 4 のエッジノードの存在を確認すると、そのことを ID 16 のコアノードを経由して ID 4 のエッジノードへ通達する。ID 16 のコアノードに TCP セッションを確立している ID 4 のエッジノードは、ID 8 のコアノードに対して TCP セッションを確立し、今まで ID 16 に対して確立していた TCP セッションは予備用となり、ID 24 に対して確立していた予備用 TCP セッションは終了する。

コアノードが離脱する例を図 4.4 を用いて示す。ID 8 のコアノードが離脱するものとする。ID 8 のコアノードは子の ID 4 のエッジノードに対して TCP セッションの確立先の変更を要請する。これにより ID 4 のエッジノードは今まで ID 16 に対して確立していた予備用 TCP セッションを本来の TCP セッションとし、ID 24 のコアノードに



Key	Value
4	16のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

Key	Value
4	16のIPアドレス
12	16のIPアドレス
16	16のIPアドレス
24	24のIPアドレス

図 4.5: エッジノード参加時

対して予備用 TCP セッションを確立し、今まで ID 8 に対して確立していた TCP セッションは終了する。そしてルーティングテーブルから自分の情報を削除し、ELA-VPN から離脱する。

エッジノードが参加する例を図 4.5 を用いて示す。ELA-VPN へ新規に ID 12 のエッジノードが参加するものとする。ID 12 のエッジノードは任意のコアノードに対して、自分はどのコアノードに TCP セッション、予備用 TCP セッションを確立すべきかを問い合わせ、実際に TCP セッションを確立する。そして親になったコアノードはエッジノードをルーティングテーブルに追加する。

4.3 モジュール構成

ELA のモジュール構成図を図 4.6 に示す。影の付いたモジュールが ELA によって提供される機能で、点線より上がユーザレイヤのソフトウェアとして動作、下がカーネルレイヤで動作する。

あるアプリケーションが他のノードと ELA-VPN 経由で通信する時、モジュール間の連携は次のようになる。アプリケーションが ELA-VPN におけるノード宛ての IP パケットを送信すると、NPD (Network Pseudo Device) に IP パケットが渡される。カ

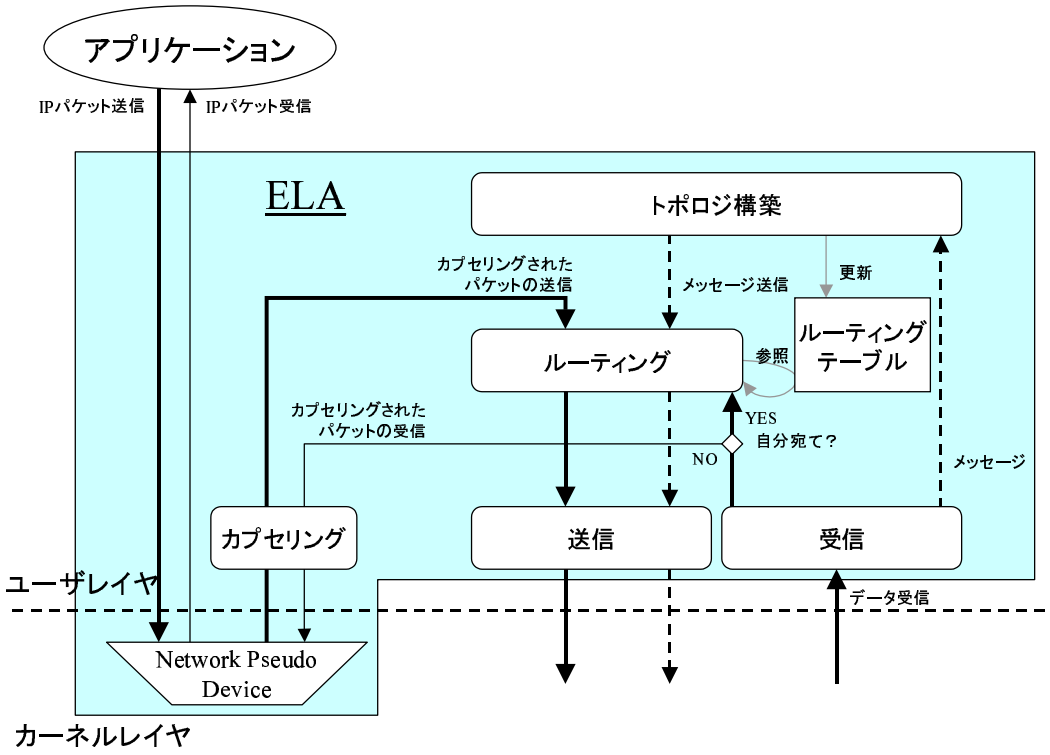


図 4.6: ELA のシステム構成

プセリングモジュールは NPD から IP パケットを取得する。カプセリングモジュールは ELA ヘッダを付加して IP パケットを暗号化を行い、ルーティングモジュールに渡す。ルーティングモジュールはルーティングテーブルを検索してデータの転送先ノードを決定し、送信モジュールに渡す。送信モジュールは受け取ったデータを、ルーティングモジュールからの指示に従って他のノードへ転送する。

反対に ELA-VPN 経由でカプセリングされた IP パケットを受信する流れは次のようになる。他のノードからカプセル化されたパケットを受信した受信モジュールは ELA ヘッダを参照して自分宛だと判断するとカプセリングモジュールに転送する。カプセリングモジュールはヘッダの削除後に復号化を行い、NPD に渡す。NPD は IP ヘッダの宛先ポートを参照して適切なアプリケーションに転送する。

各モジュールの概要を以下に示す。

Network Pseudo Device

NPD はネットワーク仮想デバイスで、ノードが ELA-VPN を経由して通信する場合のネットワークデバイスとして利用される。NPD は OS から通常のネットワークデバイスと同じように扱われ、IP アドレスやネットマスクなどを割り当て可能である。各ノードに割り当てられる ELA-IP アドレスも、NPD に設定される。

通常のネットワークデバイスは NIC (Network Interface Card) からパケットを受信

し、NIC へパケットを送信する。それに対し、NPD は後述するカプセリングモジュールからパケットを受信し、カプセリングモジュールへパケットを送信する。

トポロジ構築モジュール

ELA-VPN における P2P ネットワークの形成および維持を行う。トポロジ構築モジュールは多くの機能を持つ。

P2P ネットワークを形成するため、以下の機能を持つ。

- ユーザ認証

  悪意あるノードが成りすましをして ELA-VPN に参加することを防止するため、ノードの ELA-VPN 参加時に行うユーザ認証の機能を提供する。

- ELA-VPN における IP アドレスの決定

  ユーザ認証によって ELA-VPN への参加が妥当だとされたノードに対し、ELA-VPN におけるプライベート IP アドレスを割り当てる。新規参加するノードが特定のプライベート IP アドレスを利用したい場合、その IP アドレスが既に利用されていないかルーティングテーブルを参照して調べる。もしその IP アドレスが利用されていなければ、新規参加するノードに対してその IP アドレスが利用できることを伝達する。すでにその IP アドレスが利用されていた場合、もしくは特に利用したい IP アドレスがない場合、ルーティングテーブルを参照して利用されていないプライベート IP アドレスの中からランダムに決定し、その IP アドレスを新規参加するノードに対して伝達する。

- ノードの分類

  IP アドレスが決定した新規参加ノードをコアノードかエッジノードに分類する。新規参加ノードに対して TCP セッションの確立、UDP データグラムの送信を行い、その試行が共に成功した場合はその新規参加ノードをコアノードとする。それ以外の場合はその新規参加ノードをエッジノードとする。

P2P ネットワークを維持するため、以下の機能を持つ。

- 他ノードとのメッセージの送受信

  他ノードと情報などを交換するため、メッセージを用いた情報の共有を行う。メッセージは XML (eXtended Markup Language) 形式でやりとりされる。XML 形式を利用する利点として、メッセージのフォーマットの拡張が容易なこと、XML を解析するパーサの実装がすでに存在することが挙げられる。

- ルーティングテーブルの更新

  ELA-VPN でのノードの参加および離脱が発生するたびに、トポロジ構築モジュールはルーティングテーブルの更新を行う。

ルーティングモジュール

カプセリングモジュールあるいは受信モジュールから渡されたデータの転送先を決定する。転送データのELAヘッダを参照し、転送先ノードをルーティングテーブルより決定する。このモジュールはコアノードのみ使用する。

ELAヘッダにはELA-VPNにおける宛先IPアドレスが格納されている。ルーティングテーブルには各ノードの情報が、ELA-VPNにおけるIPアドレスを key として、インターネットにおけるIPアドレスを $value$ として格納されている。ルーティングモジュールはELAヘッダの宛先IPアドレスを key として、ルーティングテーブルから検索を行う。

検索の結果、宛先IPアドレスが他のノードだった場合、そのIPアドレスのノードへ転送するよう送信モジュールへ伝達する。宛先IPアドレスが自身だった場合、そのIPアドレスを使用している子のエッジノードへ転送するよう送信モジュールへ伝達する。ルーティングテーブルから該当するエントリが発見されなかった場合、宛先不明のデータとして破棄する。

カプセリングモジュール

ELAヘッダの処理と通信の暗号化・復号化を行う。NPDからIPパケットを渡された場合、IPパケットの先頭にELAヘッダを付加し、ELAのバージョン情報やELA-VPNにおける宛先IPアドレスなどをヘッダに格納する。次にIPパケットをトポロジ構築モジュールから渡された共通鍵を用いて暗号化を行う。

受信モジュールからカプセリングされた通信内容を渡された場合、ELAヘッダを除去し、共通鍵を用いて復号化する。そしてアプリケーションがIPパケットを受信するよう、NPDへIPパケットを渡す。

送信モジュール

カプセリングされたパケットを、ルーティングモジュールから指示されたノードへ転送する。他ノードへの転送プロトコルとしてコアノードはTCPあるいはUDP、エッジノードはTCPを用いる。コアノードは転送先ノードに応じて転送プロトコルの使い分けを行う。

受信モジュール

他ノードからカプセリングされた通信内容やメッセージを受信する。メッセージはトポロジ構築モジュールへ渡し、カプセリングされた通信内容は宛先IPアドレスが自分であればカプセリングモジュールへ、そうでなければルーティングモジュールへ渡す。

コアノードは他のノードからTCPセッションを確立される。TCPセッション確立の待機も受信モジュールによって行われる。

第5章　ELAの実装

本章ではELAの実装について述べる。最初にノード間通信の取り決めの定義を説明する。次に前章で述べたモジュールの詳細を述べ、最後にモジュールの動作の流れを説明する。

ELAの実装環境としてOSはRed Hat Linux 9.0 (Linux Kernel 2.4) を用いた。また実装言語としてC言語を用いた。

5.1　定義部分

本小節ではELA-VPNにおいて通信内容の転送に必要なELAヘッダと、ノード間において要求・応答を行うためのメッセージの定義を説明する。

5.1.1　ELAヘッダ

ELAヘッダには図5.1が示すように5つのフィールドが存在する。

- *Version*

  ELAのバージョンが格納され、フィールド長は8ビットである。本論文で提案しているELAの設計はバージョン1であるため、1の値が代入される。

- *Type*

  ELAヘッダに続くデータの種類が格納され、フィールド長は8ビットである。データの種類がカプセリングされたIPパケットの場合は`TYPE_EPACKET`, メッセージの場合は`TYPE_MESSAGE`の値が格納される。

- *TTL*

  TTL (Time To Live) フィールドはIPパケットが転送される限界の回数が格納され、フィールド長は8ビットである。ノードからノードへ転送されるごとにTTLフィールドの値は1減算される。これによって、ルーティングテーブルの不具合によりIPパケットがELA-VPNにおいて永久に転送され続けることに対処する。

- *Reserved*

  今後の拡張に備えた予約済みフィールドで、フィールド長は8ビットである。

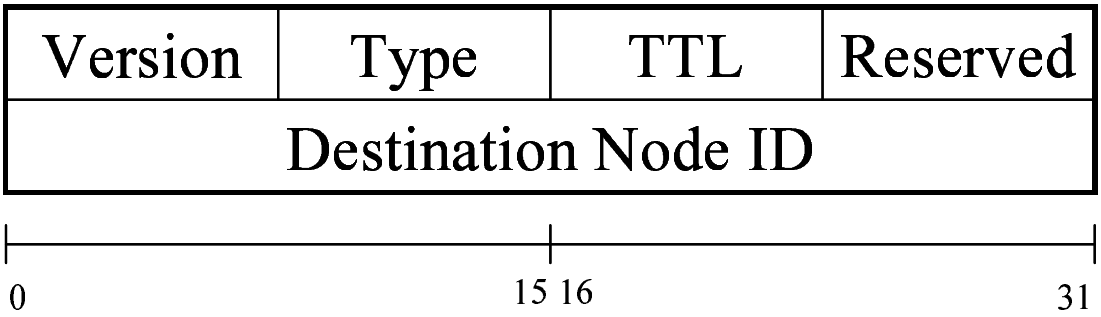


図 5.1: ELA ヘッダ

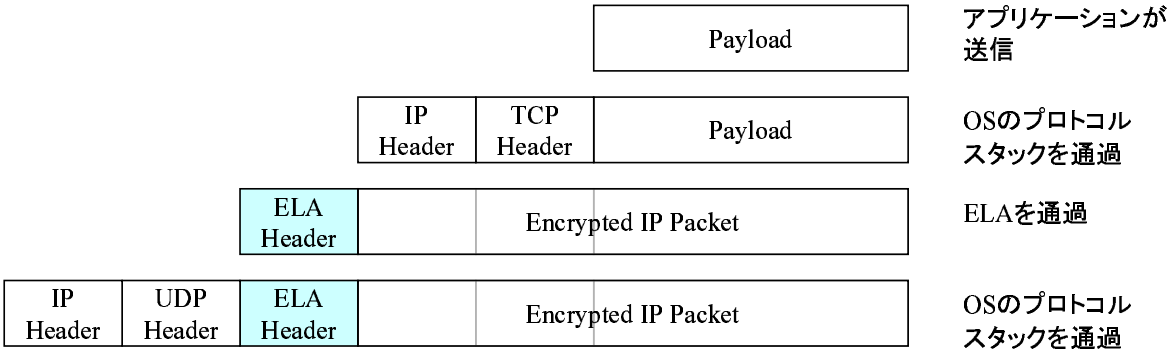


図 5.2: ELA ヘッダの位置

- *Dstination Node ID*

  ELA-VPN における宛先ノードの ID が格納され、フィールド長は 32 ビットである。

例として、TCP を用いるアプリケーションの通信データをコアノードとコアノード間で転送された場合の通信内容の遷移を図 5.2 に示す。

アプリケーションが送信したデータは TCP/IP プロトコルスタックを通過し、TCP ヘッダおよび IP ヘッダが先頭に付加される。ELA を通過することにより ELA ヘッダが先頭に付加され、これまでの IP パケットは暗号化される。カプセリングされた IP パケットは UDP/IP プロトコルスタックを通過して UDP ヘッダおよび IP ヘッダが付加された後、他のノードへ転送される。

5.1.2 メッセージ

ノード間で交換されるメッセージは XML (eXtended Markup Language) 形式である。XML 形式を用いる利点としてメッセージフォーマットの変更や拡張が容易であること、XML を解析する既存のパーサが存在することが挙げられる。

メッセージのフォーマットを図 5.3 に示す。

```
1: <?xml version="1.0"?>
2: <message>
3:   <purpose>本メッセージの目的</purpose>
4:   <type>要求か応答か？</type>
5:   <sequence>シーケンス番号</sequence>
6:   <option>オプションデータ</option>
7: </message>
```

図 5.3: メッセージのフォーマット

各行について説明を加える。1 行目では本メッセージが XML バージョン 1.0 形式であることを宣言している。2 行目ではルートノードである message の開始タグが記述されている。メッセージにおける全ての要素はこの message ノードに属する。3 行目では purpose 要素が記述されている。purpose 要素にはメッセージの目的を格納する。4 行目では type 要素が記述されている。type 要素にはこのメッセージが要求である場合は Request、応答である場合は Replya が格納される。5 行目では sequence 要素が記述されている。sequence 要素には他のメッセージと区別するための識別番号が格納される。6 行目では option 要素が記述されている。option 要素はメッセージの目的や種類によって格納されるデータが異なる。実際に格納される要素については、各モジュールの説明時に述べる。7 行目では message の終了タグが記述されており、本メッセージはこの行で終了する。

実際のメッセージは 7 行以上になる。また、本メッセージ形式は今後の ELA の設計や実装に応じて変更・拡張される可能性がある。

5.2 各モジュールの詳細

5.2.1 Network Pseudo Device

NPD はネットワークデバイスであるため、カーネルレイヤで実装する。また、実装環境で用いる Red Hat Linux 9 の Linux 2.4 カーネルは動的にデバイスを load, unload 可能な LKM (Loadable Kernel Module) 機構を備えている。LKM はカーネルの再コンパイルや再起動なしに OS にデバイスを組み込むことができる。そのため ELA の導入の手間が削減されることが期待できるので、NPD は LKM として実装を行う。

NPD は以下の方針で実装する。ユーザレイヤから open システムコールを呼ばれると、必要な初期化処理を行ってネットワークデバイスとして有効になる。反対に、ユー

Destination	Gateway	Genmask	Flags	Iface
10.0.0.0	*	255.255.255.0	U	ela0
133.27.xx.0	*	255.255.254.0	U	eth0
127.0.0.0	*	255.0.0.0	U	lo
default	133.27.xx.yy	0.0.0.0	UG	eth0

図 5.4: ルーティングテーブルの設定

ザレイヤから close システムコールを呼ばれると、必要な終了処理を行ってネットワークデバイスとして無効になる。read システムコールが呼ばれると、他のネットワークソフトウェアが送信した IP パケットをユーザレイヤへ受け渡す。反対に write システムコールが呼ばれると、ユーザレイヤから受け取った IP パケットを受信する。

Linux におけるネットワークデバイスは *net_device* 構造体によって記述される。ネットワークデバイス名を ela0 とするため、*net_device* 構造体の *name* フィールドに*"ela%d"* という文字列を格納する。また ELA-VPN では独自のルーティング手段が用いられるため、ARP (Address Resolution Protocol) を使用する必要がない。そのため NPD の初期化時に ARP および ARP のキャッシュを無効化する。

自身のアプリケーションが他のノードへ送信する場合、NPD に割り込みがかかり、IP パケットはソケットバッファのキューへ格納される。カプセリングモジュールから read システムコールを実行されると対応する ela_read 関数が実行される。ソケットバッファからキューを取り出し、copy_to_user 関数でユーザレイヤに IP パケットを引き上げる。

自身宛ての IP パケットを受信する場合、カプセリングモジュールから write システムコールを実行されると対応する ela_write 関数が実行される。ela_write 関数は copy_from_user 関数で IP パケットを取得し、eth_type_trans 関数でパケットタイプの判別を行う。そして netif_rx 関数でソフトウェア割り込みを発生させ届いたパケットを OS のプロトコルスタックのキューに登録する。

ELA の起動時は以下のような設定を行う。ELA-VPN は仮想的なネットワークセグメントであるため、各ノードには ELA-VPN におけるプライベート IP アドレスが割り当てられる。ELA-VPN のネットワークアドレスを 10.0.0.0、サブネットマスクを 255.255.255.0 とする場合、OS のルーティングテーブルを図 5.4 のように設定する。ela0 は NPD のネットワークインタフェース名である。

また NPD はネットワークデバイスであるため、ifconfig コマンドを実行することによってネットワークデバイスの一つとして表示される。ioctl システムコールを利用することにより、NPD に対して次の設定を行う。

- IP アドレス

ELA-VPN において割り当てられた IP アドレスを設定する。IP アドレスを決定する手法は第 5.2.7 節のトポロジ構築モジュールで述べる。

- ネットマスク

  ELA-VPN のネットワークセグメントのネットマスクを設定する。

- ブロードキャストアドレス

  ELA-VPN のブロードキャストアドレスを設定する。ブロードキャストアドレスは IP アドレスとネットマスクから計算により求める。

- MTU (Maximum Transfer Agent)

  Ethernet のネットワークデバイスは通常 MTU が 1500 に設定されている。これは Ethernet フレームのサイズが最大 1518 バイトと規格で決められており、フレームの先頭に付加される Ethernet ヘッダの 14 バイト、およびフレームの最後に付加される FCS (Frame Check Sequence) の 4 バイトを除いた値が 1500 バイトになるためである。

  ELA はネットワークアプリケーションが送信する IP パケットを ELA ヘッダを用いてカプセリングし、TCP/IP もしくは UDP/IP で他のノードへ転送する。そのため ELA ヘッダ、TCP ヘッダもしくは UDP ヘッダ、IP ヘッダが余分に付加されるため、余分に付加されるヘッダの総サイズだけ MTU を 1500 から小さくする必要がある。MTU を小さくしないと Ethernet フレームが 1500 バイトを超えるため、フレームがフラグメント化されてフレームの送信回数が増加し、スループットが減少する。

  TCP ヘッダは TCP オプションを用いない場合に 20 バイト、UDP ヘッダは 8 バイトであるため TCP ヘッダの方が大きい。ELA ヘッダは 8 バイト、IP ヘッダは IP オプションを用いない場合に 20 バイト従って NPD は以下の計算式により、1452 バイトに設定するのが最適である。

$$1452 = 1500 - 8 - 20 - 20$$

5.2.2 カプセリングモジュール

カプセリングモジュールは NPD から転送された IP パケットのカプセリング、および NPD へ転送するカプセリングされたパケットのカプセリング除去を行う。カプセリングモジュールは `capsulate`, `encapsulate` の 2 関数から成り立つ。

`capsulate` 関数は以下の流れにそってカプセリング処理を行う。

1. アプリケーションが NPD に送信した IP パケットを `read` システムコールで取得

   NPD はカーネルレイヤに存在するモジュールであるため、通常の関数を用いて IP パケットを取得することは出来ない。そのため NPD が提供する `read` システムコールを用いて IP パケットを取得する。

2. IP パケットの先頭に ELA ヘッダを付加

   第 5.1.1 節で述べた ELA ヘッダを付加する。ELA ヘッダには 5 つのフィールドが存在するが、以下のようにフィールドに値を代入する。*Version* フィールドには 1 を代入する。*Type* フィールドには `TYPE_EPACKET` の定数を代入する。*TTL* フィールドには 3 を代入する。本論文における設計において通信内容は 4 回以上転送されないためである。*Reserved* フィールドには 0 を代入する。*Destination Node ID* フィールドには IP ヘッダの宛先アドレスフィールドにハッシュをかけた値を代入する。次の処理において IP パケットは暗号化されるため、宛先 IP アドレスが参照できなくなる。そのため暗号化されない ELA ヘッダに宛先 IP アドレスを代入する必要がある。

3. 共有鍵を用いて IP パケットを暗号化（ELA ヘッダを除く）

   トポロジ構築モジュールから取得した共通鍵に基づき Blowfish [5] アルゴリズムを用いて暗号化する。Blowfish アルゴリズムは他の暗号化アルゴリズムに比べて処理が高速で、暗号化前と暗号化後のデータのサイズが等しいという特長を持つ。暗号化によってデータのサイズが増大するアルゴリズムを用いると、通信内容によっては通信内容のサイズが NPD の MTU を超えてフラグメントが発生する可能性があり、転送速度の低下につながる。

4. ルーティングモジュールに転送

   ルーティングモジュールが提供する `routing` 関数を用いてルーティングモジュールに転送する。

`encapsulate` 関数 は以下の流れにそってカプセリングの除去を行う。

1. ELA ヘッダの除去

   実際には ELA ヘッダを除去せず、C 言語におけるポインタを ELA ヘッダの直後に移動することで対応する。

2. 共有鍵を用いて IP パケットを復号化

   `capsulate` 関数と同様にトポロジ構築モジュールから取得した共通鍵に基づき、Blowfish [5] アルゴリズムを用いて復号化する。

3. IP パケットを `write` システムコールを用いて NPD へ転送

   NPD が提供する `write` システムコールを用いて NPD へ IP パケットの転送を行う。

5.2.3 ルーティングテーブル

ルーティングテーブルは ELA-VPN に参加するノードの ID とそれに対応するインターネットにおける IP アドレスが格納される。そのためルーティングテーブルは各ノードで同期する必要がある。

今回の実装ではルーティングテーブルにDHT (分散ハッシュテーブル) のChord [16] を用いる。Chord において key/value を保有するノードはハッシュ関数によって偏りなく分散される。key をもとに検索を行う場合、key を Chord のネットワークに送信すると、key に対応する Value を保有するノードへルーティングされる。そしてその value を保有するノードは key を送信したノードへ返信することにより、key に対応した value を参照できる。

Chord は以下の API を提供する。

- `insert(key, value)`

  key/value を DHT に追加する。

- `lookup(key)`

  key に対応する Value を DHT から検索して返す。

- `update(key, newvalue)`

  key/value を更新する。

- `join()`

  Chord ネットワークに自身のノードを追加する。

- `leave()`

  Chord ネットワークから自身のノードを削除する。

コアノードは ELA-VPN に参加すると最初に `join` 関数を実行し、自身を Chord ネットワークに追加する。ELA-VPN にノードが参加した場合、トポロジ構築モジュールが `insert` 関数を実行し、エントリを追加する。ELA-VPN からノードが離脱した場合、トポロジ構築モジュールが `update` 関数を実行し、エントリを削除する。ルーティングモジュールがルーティングテーブルを参照する場合は `lookup` 関数を実行し、ノードの ID に対応するインターネットにおける IP アドレスを取得する。コアノードが ELA-VPN から離脱する場合は `leave` 関数を実行し、自身を Chord ネットワークから削除する。

5.2.4 ルーティングモジュール

ルーティングモジュールはカプセリングされた IP パケットをルーティングテーブルを参照することによって次に転送すべきノードを決定し、送信モジュールへ渡す。エッジノードはルーティング処理をしないため、ルーティングモジュールでは送信モジュールへの転送のみ行う。

ルーティングモジュールによって提供される `routing` 関数は以下の流れでルーティング処理を行う。

コアノードの場合

1. 宛先 IP アドレスを取得

   ELA ヘッダを参照することにより ELA-VPN における宛先 IP アドレスを取得する。

2. ルーティングテーブルを参照

   ルーティングテーブルの API である `lookup` 関数を実行し、インターネットにおける IP アドレスを取得する。

3. 送信モジュールへ転送

   転送先ノードがコアノードであれば `ela_udpsend` 関数、転送先ノードがエッジノードであれば `ela_tcpsend` 関数を実行し、送信モジュールへカプセリングされたパケットを渡す。

エッジノードの場合

親のコアノードと確立している TCP セッションのファイルディスクリプタを引数に、送信モジュールが提供する `ela_tcpsend` 関数を実行する。`routing` 関数の擬似コードを図 5.5 に示す。

5.2.5 送信モジュール

送信モジュールはルーティングモジュールに指定されたノードへデータを転送する。送信モジュールは `ela_tcpsend` 関数と `ela_udpsend` 関数を提供し、共にルーティングモジュールから呼び出される。

`ela_tcpsend` 関数はルーティングモジュールに指定された TCP セッションのファイルディスクリプタを引数に指定し、`send` システムコールを用いて他のノードへ転送する。`ela_udpsend` 関数はルーティングモジュールに指定されたインターネットにおける IP アドレスを引数に指定し、`sendto` システムコールを用いて UDP データグラムを他のノードへ転送する。

今回の実装において `ela_tcpsend` 関数および `ela_udpsend` 関数は、呼ばれると単純に他ノードへデータの転送のみを行う。そのため実際の ELA の利用において以下の問題が想定される。

- `ela_tcpsend` 関数における問題

  TCP は途中経路で紛失した通信内容の再送、および輻輳制御を行う。そのため TCP による通信内容を TCP で転送する場合、途中経路で通信内容を紛失すると一つの通信内容に対し TCP の再送制御が二度行われるという TCP over TCP [18] の問題が発生する。これにより、使用可能帯域が大きく減少する。

```
routing(epakcet){
  if(me == CORENODE){
    dst_ipaddr = epacket.DST_IPADDR;
    nextnode = lookup(dst_ipaddr);
    if(nextnode == CORENODE){
      ela_udpsend();
    }else if(nextnode == EDGENODE){
      ela_tcpsend();
    }else{
      return FALSE;
    }
  }else if(me == EDGENODE){
    ela_tcpsend();
  }else{
    return FALSE;
  }
  return TRUE;
}
```

図 5.5: routing 関数の擬似コード

- `ela_udpsend` 関数における問題

  UDP は途中経路で紛失した通信内容の再送を行わない。そのため ELA が想定環境としているインターネットにおいての利用では、LAN 環境に比べ通信内容が紛失しやすい。通信内容の紛失がほとんど発生しない LAN を想定した NFS などのアプリケーションは正常に利用できない可能性がある。

- 二つの関数に共通する問題

  QoS (Quality of Service) を考慮していない。そのためあるエッジノードが大量の通信を行うことにより、親のコアノードの利用可能帯域が大幅に減少するということが想定される。ユーザの要求に従い、ELA で使用する帯域をある一定以下に抑えるなどの処理が必要である。

これらは今後の課題とする。

```
ela_recv(data){
  type = data.TYPE;
  if(type == TYPE_MESSAGE){
    msg_recv(data);
  }else if(type == TYPE_EPACKET)
    dst_ipaddr = data.DST_IPADDR;
    if(dst_ipaddr == me){
      encapsulate(data);
    }else{
      routing(data);
    }
  }else{
    return FALSE;
  }
  return TRUE;
}
```

図 5.6: ela_recv 関数の擬似コード

5.2.6 受信モジュール

受信モジュールは他ノードから転送されたデータを受信し、データの ELA ヘッダを参照してトポロジ構築モジュール、ルーティングモジュールあるいはカプセリングモジュールのいずれかに渡す。受信モジュールは ela_tcprecv 関数、ela_udprecv 関数および ela_recv 関数を提供する。

ela_tcprecv 関数は他ノードと確立している TCP セッションからデータを受信し、ela_recv 関数へ渡す。ela_udprecv 関数は UDP データグラムを受信し、ela_recv 関数へ渡す。このように ela_tcprecv 関数と ela_udprecv 関数は他ノードからのデータ受信のみを行い、データの判別処理は次に説明する ela_recv 関数によって行われる。

ela_recv 関数は最初に受信したデータの ELA ヘッダを参照する。ELA ヘッダの *TYPE* フィールドが TYPE_MESSAGE の場合、ela_recv 関数はデータをメッセージとみなし、トポロジ構築モジュールへ渡す。ELA ヘッダの *TYPE* フィールドが TYPE_EPACKET の場合、ela_recv 関数はデータをカプセリングされたパケットとみなす。もし宛先 IP アドレスが自身の IP アドレスと等しい場合はカプセリングモジュールにデータを渡して受信する。そうでなければルーティングモジュールへ渡し、他のノードへ転送する処理を行う。

ela_recv 関数の擬似コードを図 5.6 に示す。

表 5.1: メッセージの種類

purpose 要素	目的
NodeAuthentication	ノードのユーザ認証
NodeUsingPort	ノードの利用するポートの伝達
NodeClassfication	ノードの分類
NodeJoin	ノード参加の伝達
NodeLeave	ノード離脱の伝達
ChangeParent	親のコアノードの変更要求

5.2.7 トポロジ構築モジュール

トポロジ構築モジュールは ELA-VPN の P2P ネットワークの形成および維持を行う。P2P ネットワークの形成機能としてユーザ認証と ELA-VPN における IP アドレスの決定、ノードの分類を含む。P2P ネットワークの維持機能としてメッセージを用いたノードの参加離脱情報の共有、ルーティングテーブルの更新を含む。

今回はトポロジ構築モジュールを実装していないため、本小節では実装方針について述べる。

メッセージの送受信

受信モジュールはメッセージを受信するとトポロジ構築モジュールの `msg_recv` 関数を実行する。`msg_recv` 関数は受け取った XML 形式のメッセージを解析する。メッセージは様々な目的に応じて利用される。今回の実装におけるメッセージの目的の一覧を表 5.1 に示す。

ユーザ認証と IP アドレスの決定

ELA-VPN に悪意ある人物を参加させないため、ELA-VPN に新規に参加しようとするノードに対しユーザ認証を行う。

例として次のような状況を想定する。コアノードのノード A は ELA-VPN に既に参加しており、ノード S はこれから ELA-VPN に参加しようとしている。ノード S は sada というユーザ名で、ノード A を通じて ELA-VPN に参加する。

ノード S は既にノード A へ TCP セッションを確立し、図 5.7 のようなメッセージを送信する。

ノード A はノード S に対し RSA 認証アルゴリズムを用いたユーザ認証を行う。RSA 認証に必要とされる公開鍵は ELA によって生成される。

```
<?xml version="1.0"?>
<message>
  <purpose>NodeAuthentication</purpose>
  <type>Request</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <username>sada</username>
  </option>
</message>
```

図 5.7: ユーザ認証要求のメッセージの例

```
<?xml version="1.0"?>
<message>
  <purpose>NodeUsingPort</purpose>
  <type>Request</type>
  <sequence>xxxxxxx</sequence>
  <option></option>
</message>
```

図 5.8: 利用ポート要求のメッセージの例

ノードの分類

ユーザ認証が妥当と判断されたノードをコアノードとエッジノードに分類する。前小節と同様、既に ELA-VPN に参加しているノード A を通じてノード S が ELA-VPN に参加すると想定する。

ノード A はノード S に対してどのポートで TCP および UDP の通信を行うかを図 5.8 のようなメッセージを送信して訊ねる。

ノード S はノード A に対して図 5.9 のようなメッセージを送信することにより、指定したポートへ TCP セッションを確立し、UDP データグラムを送信するように伝達する。

ノード A は受信したメッセージに従い、ノード S の TCP 10000 ポート, UDP 20000 ポートへデータ送信を試みる。成功すればノード A はノード S をコアノードと認定し、図 5.10 のメッセージを伝達する。

```
<?xml version="1.0"?>
<message>
  <purpose>NodeUsingPort</purpose>
  <type>Reply</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <tcp>
      <port>10000</port>
    </tcp>
    <udp>
      <port>20000</port>
    </udp>
  </option>
</message>
```

図 5.9: 利用ポート応答のメッセージの例

ELA-VPN からの離脱

- コアノードの場合

  まず子のエッジノードに対し、親のコアノードを変更するようメッセージを送信する。例えば新しい親のコアノードが 10.0.0.100 である場合は図 5.11 のようなメッセージを送信する。

  子のエッジノードはメッセージを受信すると、現在の TCP セッションを確立しなおして親のコアノードを変更する。最後にルーティングテーブルの API である `leave()` 関数および `update` を実行し、自分自身のエントリを削除する。

- エッジノードの場合

  例えば ELA-VPN におけるプライベート IP アドレスが 10.0.0.100 のエッジノードが離脱する場合、親のコアノードに対して図 5.12 のようなメッセージを送信する。

  このメッセージを受信したコアノードはルーティングテーブルから IP アドレス 10.0.0.100 のエントリを削除し、エッジノードに対して ELA-VPN 離脱の許可を与える図 5.13 のようなメッセージを送信する。

  このメッセージを受信したエッジノードは親に対して確立している TCP セッションを切断し、ELA-VPN から離脱する。TCP セッションを切断された親のコアノードは自身に対して確立されている TCP セッションのリストを更新する。

```
<?xml version="1.0"?>
<message>
  <purpose>NodeClassification</purpose>
  <type>Request</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <nodetype>CoreNode</nodetype>
  </option>
</message>
```

図 5.10: ノード分類結果メッセージの例

```
<?xml version="1.0"?>
<message>
  <purpose>ChangeParent</purpose>
  <type>Request</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <node>10.0.0.100</node>
  </option>
</message>
```

図 5.11: 親のコアノード変更要求のメッセージの例

しかしノードの故障やネットワークからの突然の切断などにより、上記の正規の離脱手順を経ずに ELA-VPN から離脱するノードが想定される。あるノードに対して n 回連続して通信に失敗する場合にはそのノードは ELA-VPN から離脱したものとみなす。トポロジ構築モジュールを実装後、n の値はいくつが最適であるかを評価して決定する。このような処理を行うことによってルーティングテーブルを維持する。

ルーティングテーブルの更新

受信モジュールから他のノードの参加離脱を伝達されるとそれに応じてルーティングテーブルを更新する。例えば、エッジノードは離脱する場合に親のコアノードに対しメッセージを用いて伝達する。そのメッセージを受け取ったコアノードはルーティングテーブルの API である update 関数を実行し、ルーティングテーブルを更新する。

```
<?xml version="1.0"?>
<message>
  <purpose>NodeLeave</purpose>
  <type>Request</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <ipaddr>10.0.0.100</ipaddr>
  </option>
</message>
```

図 5.12: ノード離脱伝達のメッセージの例

```
<?xml version="1.0"?>
<message>
  <purpose>NodeLeave</purpose>
  <type>Reply</type>
  <sequence>xxxxxxx</sequence>
  <option>
    <result>OK</result>
  </option>
</message>
```

図 5.13: ノード離脱許可のメッセージの例

5.3 モジュールの連携

本章では ELA-VPN 経由でアプリケーションがデータの送受信を行うときの、各モジュールの具体的な連携を説明する。

5.3.1 送信時の流れ

アプリケーションが ELA-VPN の他のノードへデータを送信した場合の、モジュールの動作の流れを図 5.14 に示す。

アプリケーションが ELA-VPN のプライベート IP アドレス宛の通信を行った場合、NPD に割り当てられた IP アドレスと OS のルーティングテーブルにより NPD に IP パケットが渡される。カプセリングモジュールは `capsulation` 関数内で `read` システムコールを用いて NPD から IP パケットを取得し、`routing` 関数を用いてルーティングモジュールに渡す。`routing` 関数はルーティングテーブルを参照して次に転送するノー

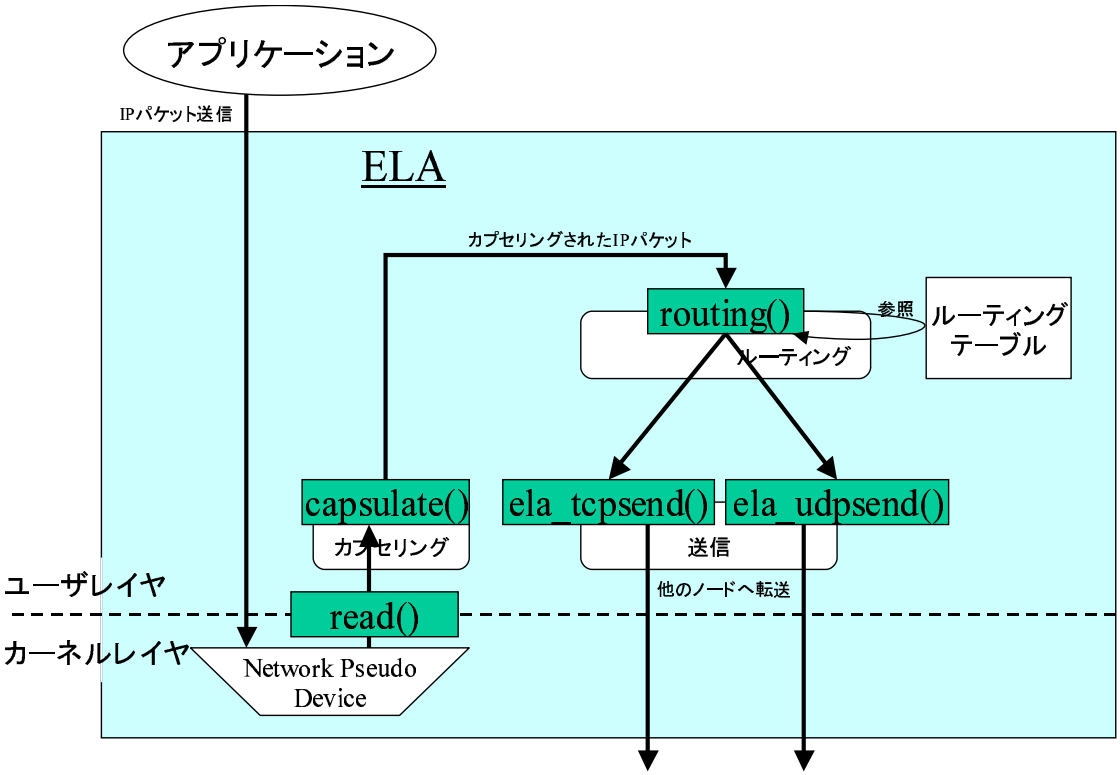


図 5.14: IP パケット送信時の流れ

ドを決定する。転送先ノードへ TCP でデータ転送する場合には `ela_tcpsend` 関数、UDP でデータ転送する場合には `ela_udpsend` 関数を呼び、他のノードへ転送する。

5.3.2 受信時の流れ

他のノードからデータを転送され、アプリケーションがデータを受信する場合のモジュールの動作の流れを図 5.15 に示す。

TCP で通信しているノードからデータを受信する場合は `ela_tcpread` 関数、UDP で通信しているノードからデータを受信する場合は `ela_udpread` 関数を呼ぶ。どちらの関数も `ela_read` 関数にデータを渡す。`ela_read` 関数はデータがカプセリングされた IP パケットで、かつ自分宛であると判断すると `encapsulation` 関数を実行し、カプセリングモジュールへ渡す。カプセリングモジュールはカプセリングの除去後、`write` システムコールを用いて NPD に渡す。NPD は IP パケットを受信し、該当するポートを使用しているアプリケーションに渡され、アプリケーションはデータを受信する。

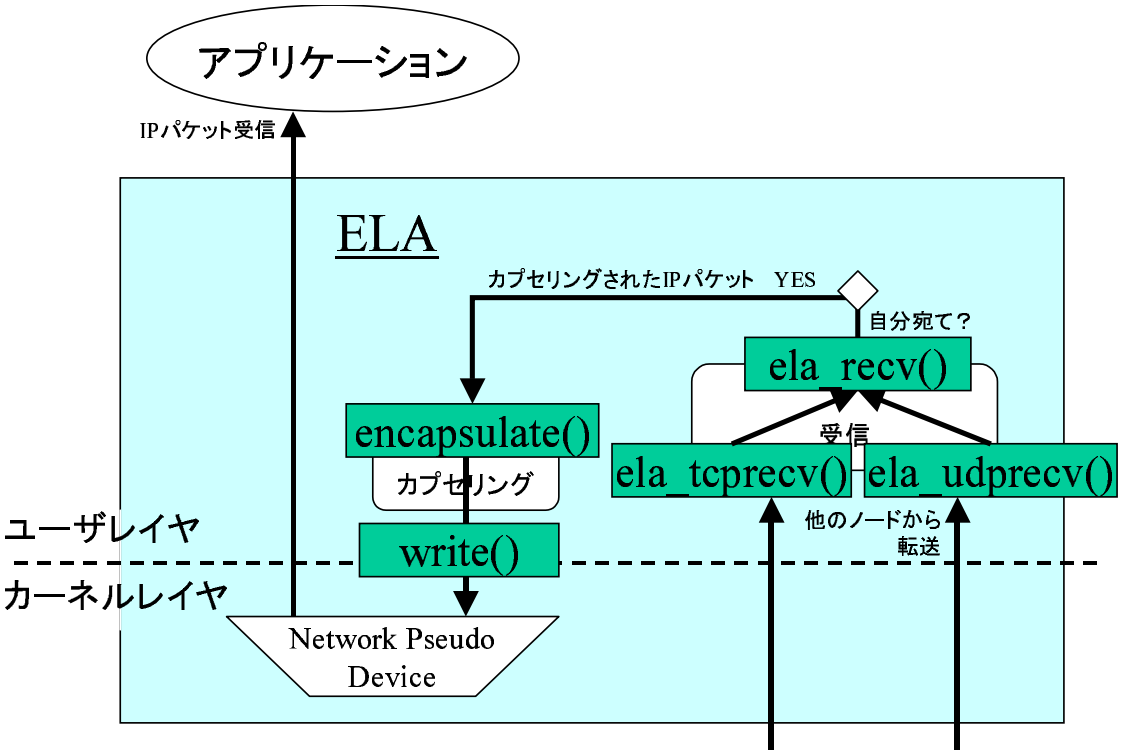


図 5.15: IP パケット受信時の流れ

第6章　評価

6.1　オーバヘッドの評価

ELA によるオーバヘッドを評価するため、ELA による通信の遅延を測定した。そのために、次の環境を用意した。ルータ (CPU: Celeron 433MHz, Memory 192MBytes, OS: Red Hat Linux 9) 下に 2 台の PC (CPU: Pentium M 1.7GHz, Memory 1GByte, OS: Red Hat Linux 9) を 100Base-TX で接続し、それぞれ異なるネットワークセグメントを割り当てた。そして以下のケースを想定し、`ping` を用いて RTT (Round Trip Time) を 512 回測定した。

1. ELA-VPN を構築して 2 ノードともコアノードの場合 (UDP による転送)
2. ELA-VPN を構築してそれぞれコアノードとエッジノードの場合 (TCP による転送)
3. ELA-VPN を利用しない場合の RTT

表 6.1: 2 ノード間における遅延 (単位 msec)

	平均	標準偏差
1. コアノード同士	3.74	2.91
2. コアノードとエッジノード	4.02	2.55
3. ELA なし	0.22	0.04

表 6.1 は **1.** と **2.** が **3.** のケースよりも遅延の平均が大きく、かつ分散しており、**1.** と **2.** に大きな違いはないことを示している。遅延の増加の原因として ELA によるオーバヘッドが存在することが挙げられる。ELA はカーネルレイヤの通信内容を一度ユーザレイヤに引き上げ、ルーティングや暗号化処理等を施してから再びカーネルレイヤに戻すため、オーバヘッドが生じて遅延が発生する。

第7章　結論

本章では本論文のまとめと今後の課題について述べる。

7.1　本論文のまとめ

本論文では異種ネットワークセグメントに属するノード間で P2P ネットワーク型の VPN の構築を実現する ELA を提案した。ELA はインターネットに接続するユーザのノード同士が直接通信する時における、セキュアな通信基盤としての VPN を構築することを目的とする。上記の目的の VPN を構築するには、従来の VPN 機構は多くの手間やコストを必要とするが、ELA は即興的に VPN を構築可能である。また ELA のプロトタイプ実装を行い、VPN によるオーバヘッドの測定を行った。

ELA を用いることにより、インターネットを介したユーザ同士のよりセキュアな通信を可能にする。昨今の技術革新によりノードの処理性能はさらに高速化し、ネットワークはさらに広帯域化することが予想される。それに伴い、ユーザのノード同士が直接通信するネットワークアプリケーションも、今後さらに多く登場すると考えられる。そのような状況下において ELA はますます重要性が高まるであろう。

7.2　今後の課題

今後の課題として以下の事項が挙げられる。

- トポロジ構築モジュールとルーティングモジュールの実装

  今回行ったプロトタイプ実装ではトポロジ構築モジュールおよびルーティングモジュールの実装を行っていない。そのため何台のノードまで実用的に動作するか、などのスケーラビリティの評価を行えない。今後はトポロジ構築モジュールおよびルーティングモジュールの実装を行い、評価を行うことによって ELA の有用性を検証する。

- QoS のサポート

  現在の設計は QoS (Quality of Service) を考慮していない。そのため、あるエッジノードが大量に通信を行うことにより親のエッジノードの利用可能帯域が大きく減少する、などの問題が想定される。また帯域の狭いノードがコアノードになることにより、子のエッジノードが利用可能な帯域も狭くなるという問題も想定

される。一定以上の帯域の通信を保証し、通信を一定以下に制限するような QoS の実現が必要である。

- コアノードの増加

  エッジノードが他のノードと通信する場合は必ず親のコアノードを経由する。そのため、例えば ELA-VPN にコアノードが 1 つしか存在しない場合はエッジノード同士の通信は全て 1 つのコアノードに通信が集中し、Client/Server 型 VPN と同様の問題が発生する。そのため NAT 越え技術などを利用し、コアノードを増加させる必要がある。

謝辞

本研究を進めるにあたり、御指導を頂きました、慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します.

間 博人氏、斉藤 匡人氏、滝沢 允氏、高橋 ひとみ氏、峰松 美佳氏をはじめとする慶應義塾大学徳田研究室 ECN 研究グループの諸氏、また徳田・村井・楠本・中村・南研究室の方々には，研究会の活動を通じて多くの御意見、御助言を頂きましたこと拝謝します。

そして研究の日々を共に過ごした、金田 裕剛氏、山下 勝司氏、その他多くの友人に感謝します。

最後に、いつも温かい声援で激励してくれたピアノサークル K'ISS の多くの友人と、今日まで絶え間ない努力によって私を育ててくれた家族に心から感謝の意を表し、謝辞と致します。

2004 年 12 月 29 日

青柳 禎矩

参照論文

- 青柳　禎矩, 滝沢　允, 斉藤　匡人, 間　博人, 徳田　英幸.
“異種セグメント端末による分散型仮想 LAN 構築機構の設計と実装, ” 情報処理学会 全国大会, vol.66 (3) pp. 557-558,
Mar. 2004.

- 青柳　禎矩, 滝沢　允, 斉藤　匡人, 間　博人, 徳田　英幸.
“ELA: 分散型 VPN の設計と実装, ” 日本ソフトウェア科学会 The Sixth Workshop on Internet Technology (WIT2004),
Dec. 2004.

- Sadanori Aoyagi, Makoto Takizawa, Masato Saito, Hiroto Aida, Hideyuki Tokuda.
“ELA: A Fully Distributed VPN System over Peer-to-Peer Network, ” IEEE International Symposium on Applications and the Internet (SAINT 2005),
Feb. 2005.

参考文献

[1] Emotion link:
http://www.freebit.com/solution/emotion.html.

[2] IP Security Protocol (ipsec) Charter,
http://www.ietf.org/html.charters/ipsec-charter.html.

[3] Lina Alchaal, Vincent Roca, and Michel Habert. Offering a Multicast Delivery Service in a Programmable Secure IP VPN Environment. *In proceedings NGC 2002*, October 2002.

[4] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, A. Malis. A Framework for IP Based Virtual Private Networks, RFC 2764, February 2000.

[5] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*. John Wiley & Sons, Inc, 1996.

[6] Mark Carson and Darrin Santay. Nist net: a linux-based network emulation tool. *In Proceedings SIGCOMM Comput. Commun. Rev.*, Vol. 33, No. 3, pp. 111–126, 2003.

[7] Daiyu Nobori. SoftEther. http://www.softether.com/.

[8] G. Pall, G. Zorn. Microsoft Point-To-Point Encryption (MPPE) Protocol, RFC 3078, March 2001.

[9] id Software. Doom.
http://www.idsoftware.com/, 1993.

[10] Jon Postel. User Datagram Protocol, RFC 768, August 1980.

[11] Jon Postel. Transmission Control Protocol, RFC 793, September 1981.

[12] K. Egevang, P. Francis. The IP Network Address Translator (NAT), RFC 1631, May 1994.

[13] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn. Point-to-Point Tunneling Protocol (PPTP), RFC 2637, July 1999.

[14] Maxim Krasnyansky. Virtual Tunnel (VTun). http://vtun.sourceforge.net/.

[15] S. Hanks, T. Li, D. Farinacci, P. Traina. Generic Routing Encapsulation (GRE), RFC 1701, October 1994.

[16] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 149–160, 2001.

[17] Olaf Titz. Crypto IP Encapsulation (CIPE). http://sites.inka.de/sites/bigred/devel/cipe.html.

[18] Olaf Titz. Why TCP Over TCP Is A Bad Idea. http://sites.inka.de/bigred/devel/tcp-tcp.html.

[19] W. Simpson. The Point-to-Point Protocol (PPP), RFC 1661, July 1994.

[20] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, B. Palter. Layer Two Tunneling Protocol "L2TP", RFC 2661, August 1999.

[21] Akio Yamamoto. TinyVPN. http://www.shimousa.com/tv/.

[22] James Yonan. OpenVPN. http://openvpn.sourceforge.net/.