NAIST-IS-MT0351040

修士論文

エンドノード主導によるファイアウォール制御機構の提案と実装

鬼丸 敬輔

2005 年 2 月 3 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

鬼丸 敬輔

審査委員：　砂原 秀樹 教授
山口 英 教授
藤川 和利 助教授

エンドノード主導によるファイアウォール制御機構の提案と実装 *

鬼丸 敬輔

内容梗概

インターネットで流通する情報は増大しており、現在のクライアント・サーバモデルではサーバ側の負担が大きな問題となっている。そのため P2P モデルが注目されているが、P2P モデルの実現にはエンドノード同士が相互に接続可能でなければならない。相互接続性を確保し、それによって P2P モデルを実現するための技術には、SOCKS や IP VPN などの仮想ネットワーク技術がある。しかしこれらの技術は、利用者を事前に設定しなければならず、またファイアウォールの保護が適切に行われない、あるいはネットワーク管理者が通信を捕捉することができないなどの問題がある。このように仮想ネットワーク技術を容易に用いることはできず、それらの技術を前提とした P2P アプリケーションは実現していない。

本論文ではこの問題点を解消するために、エンドノードの主導によってファイアウォールを動的に再構成するファイアウォール制御技術について提案する。また、ファイアウォールを制御し、通信を可能とする上で必要なネットワークやノードの安全性を維持するための方法として、エンドユーザに立脚した、エンドユーザ同士による信頼モデルについて提案する。

提案の有用性を検証するため、提案に基づいたファイアウォール制御機構を設計、実装した。また提案機構の実験と評価を行い、本提案が P2P モデルの実現に寄与することを示した。

キーワード

ファイアウォール制御, セキュリティ, 電子証明書, アクセス制御

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 修士論文, NAIST-IS-MT0351040, 2005 年 2 月 3 日.

A proposal and Implementation for End-node Oriented Firewall Traverser*

Keiuske Onimaru

Abstract

As data exchanged on the Internet incleases, what loads of server side inclease become a problem on current server-client model. Now P2P model is propsed to replace the server-client model. However, all end-nodes must be interconnectable to make up the P2P model. Technologies for keeping interconnectivity and realizing P2P model include virtual private network technology like as SOCKS, IP VPN, etc., but these technologies have some problems. To use these technologies, we must specify the user to accept beforehand, and generally end-nodes on virutal private network don't be protected by real network firewall, moreover operators of real network cannot watch the content of communication on virtual private network because any connections are encrypted. Therefore these technologies cannot be adapted to the P2P applications.

In this thesis, I proposed the end-node oritented firewall control mechanism to solve these problems. And I proposed the end user oriented mutual trust model for applying this technology. Moreover I develop a system to verify the usability of my proposal mechanism. As a result of experiments, I confirmed that the mechanism has some effectiveness.

Keywords:

Firewall Traversal, Internet Security, Certification, Access Control

*Master's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0351040, February 3, 2005.

目 次

図 目 次

表 目 次

第1章　序論

インターネットは元々、現在のように多数の利用者を抱えることを想定して設計されていなかった。1969 年に米国防総省の軍事用プロジェクトとして誕生し、以降は研究用として多数の研究機関や大学などの教育機関が接続したことで、多くの研究者や教育関係者が利用するようになったが、それでも利用者はある程度限定されていた。

しかし 1990 年代前半に商用 Internet Service Provider (ISP) が登場して以来、利用者は飛躍的に増大した。企業での業務利用が日常化し、家庭から個人が World Wide Web (WWW) の閲覧、電子メールの交換など娯楽の一つとして利用することも増えた。インターネットに接続するノードは日に日に増え、インターネットの接続ノード数は 2 億 8513 万台（2004 年 7 月現在）を超えており、10 年前の 321 万台と比較すると 88 倍にもなった [1]。これは、インターネットの誕生時には想像されていなかったことである。

一方でインターネットは、今や業務や生活に欠かすことのできない社会基盤として認知されるようになった。しかし社会基盤として求められる安定性に関しては、十分とはいえない。インターネットで用いられる TCP/IP の設計は、現在のように不特定で数億もの利用者を想定していなかったため、利用者の多様化によって増加した悪意ある通信についての対処ができず、また利用者の増加によって IP アドレスが枯渇するなど、様々な問題が生じるようになった。

悪意ある通信の増加は、利用者が多様化したことに一つの原因がある。研究機関や教育機関のみが接続していた時代には、利用者は研究者や教育関係者などがほとんどであり、良識を持っているという性善説に立つことが可能であった。しかし商業化によって利用者は多様化し、善であると仮定することは困難となった。事実、特定のノードへの Denial of Service (DoS) 攻撃や、インターネットワームといった悪意ある通信が、しばしば社会現象として報じられている。

IP アドレスの枯渇は、利用者数の急激な増加が原因である。IPv4 アドレスは 32 ビット長であり、理論上は 2^{32} である約 42 億のノードを固有に識別することができるが、実

際にはパケットの転送を効率化するために構造化されており、現在のように数億台のノードを接続することを考えた設計とはなっていない。

これらの課題は早くから問題となっており、インターネット、および IP には様々な改善が加えられてきた。一つは、悪意のある通信からネットワークやノードを保護するためのファイアウォールである。もう一つは、IP アドレスの枯渇を避けるために、1 つの IP アドレスで複数のノードをネットワークへ接続する Network Address Translation (NAT)[2] である。

これらの技術の導入によって、課題のいくつかは解消された。しかし副作用として、ノードが相互に接続可能であるという、相互接続性が保証されなくなった。当初のインターネットでは、相互接続性を前提として様々なサービスが提供されたが、これらの技術によって相互接続性が保証されなくなったことで、現在では新たにクライアント・サーバモデルが用いられるようになった。

クライアント・サーバモデルとは、接続を受け付けサービスを提供するサーバと、接続を行いサービスを受けるクライアントという 2 種類にノードを分割し、役割を固定化する接続モデルである。クライアント・サーバモデルでは、全体のノードのうちサーバのみ相互接続性を確保すればよく、相互接続性の保証されない現在のインターネットに適している。しかし利用者の飛躍的な増加や、アクセス回線の高速化によって取り扱われる情報は著しく増大し、サーバやそのサーバを接続するネットワーク、ルータの負荷が問題となっている。

そのため、当初のインターネットで用いられていた、相互接続によるサービスの提供に再び注目が集まっている。このモデルをクライアント・サーバモデルとの対比で、現在では Peer-to-Peer (P2P) モデルと呼ぶ。P2P モデルではクライアントとサーバのように役割を二分しないため、特定のノードに負荷が集中せず、効率のよいサービス提供ができる。しかし P2P モデルの実現には、当初のインターネットのように、相互接続性を欠かすことはできない。

相互接続性を確保する技術、ひいては P2P モデルを実現する技術として、ファイアウォール透過技術や仮想ネットワーク技術などが提案されてきた。しかし、これらの技術には様々な問題がある。代理サーバが接続を仲介するファイアウォール透過技術では、代理サーバの負担が大きく通信速度が著しく低下する。物理的なネットワークの上に仮

想的なネットワークを構築する仮想ネットワーク技術では、物理的なネットワークの管理者が通信を制御することが困難であり、そのため仮想ネットワークの構築は制限されることが少なくない。このため、これらの技術の利用を前提とした P2P アプリケーションは実現に至っていない。

本論文ではこれらの問題を解決するために、エンドノードの主導によってファイアウォールを動的に再構成する、ファイアウォール制御技術について提案する。本提案は、エンドノードとファイアウォールが連携することで、ファイアウォールがエンドノードの要求に従い、またエンドノードの設定やエンドユーザの能力に応じて、ファイアウォール自身を再構成することを可能とする。加えて、接続を要求する側と接続を受け入れる側の、双方のファイアウォールが相互に連携することでエンドノード間の相互接続を可能とする。

本提案は、代理サーバを用いるファイアウォール透過技術よりも高速で、ネットワーク管理者が通信を捕捉することが可能であり、開発が容易で広域に展開可能な P2P アプリケーションの実現に寄与できると考える。

本論文は、8 つの章で構成されている。本章に続く第 2 章では、インターネットの現状と、今後インターネットに望まれる要求事項について述べる。第 3 章では、既存の相互接続性を確保する技術、また P2P モデルを実現する技術について概説する。第 4 章では、それらの既存技術をふまえ、新たな相互接続性を確保するための機構について提案する。第 5 章では、提案した機構の詳細な設計を、第 6 章ではその設計に基づいた実装を示す。第 7 章では、この実装を用いた実験を行い、提案と実装について評価する。最後に第 8 章では、本研究のまとめと今後の課題を明らかにする。

第2章　現状と要求事項

本章では、インターネットがおかれている現状と、その現状をふまえて、インターネットに要求されている事項について説明する。

2.1. 悪意ある通信の増加とアドレスの枯渇

インターネットの拡大によって生じた、悪意ある通信の増加、IP アドレスの枯渇という課題の解消のため、インターネットは当初とは異なる仕組みを導入した。一つは、悪意のある通信からネットワークやノードを保護するためのファイアウォールである。もう一つは、1 つの IP アドレスで、複数のノードをネットワークへ接続するための Network Address Translation (NAT) である。

本節では現状のファイアウォールと NAT の役割と問題点について述べる。

2.1.1 ファイアウォール

ファイアウォールは、インターネットのような外部ネットワークと、企業や組織の内部ネットワークを接続する境界地点に設置し、この両者を隔離する働きを持つハードウェア、またはソフトウェアである。

もともとファイアウォールとは建築用語であり、火災が発生した際に建物内の延焼を食い止めるために、ビルやマンションといった大型建造物に設置される特殊な壁である。これによってビルの一部で火災が発生しても、他の部分へ火が回るのを遅らせたり、抑えることができる。ネットワークにおけるファイアウォールも、インターネットからネットワークやノードを隔離する壁となり、インターネット上で何かが起こった場合に、保護するネットワークやノードへの影響を最小限に留める役目を果たす。

ファイアウォールの基本的な機能は、通過するパケットの識別子を確認し、あらかじめ定められたルールに従ってパケットの通過、または破棄を決定することである。ファ

イアウォールは汎用 OS 上で動作するソフトウェアとして登場したが、現在ではネットワーク速度の向上に伴って処理速度の向上が必要となり、専用のハードウェアや IC チップも開発されている。また、機能も複雑化しており、通過と破棄を判断するルール設定も、より高度に行うことが可能となっている。

ファイアウォールの基本的な利用例は、外部ネットワークのノードから、内部ネットワークのノードへの接続、または、内部ネットワークのノードから外部ネットワークのノードへの、特定の接続を禁止することである。

前者は、外部の悪意をもった第三者から、内部ネットワークを保護する意味がある。外部から内部ネットワークのノードへ自由に接続することが可能な場合、ノードへ接続すると見せかけた攻撃によって、そのノードの制御を奪われる危険性がある。結果として、内部ネットワークに蓄積されている情報が盗まれたり、そのノードが別のネットワークへの攻撃の踏み台として利用されることがある。

後者が用いられる理由は、主に二つある。一つは、内部ネットワークの利用者による、企業や組織にとってリスクの高い通信を遮断するためである。重要な情報の漏洩を防ぐため Web の掲示板への接続を禁止したり、メールサーバへの接続を制限することができる。もう一つは、ネットワークが提供されている本来の目的とは異なる目的の通信を遮断するためである。企業などでは、業務とは無関係な Web ページへの接続を禁止することで、無駄なネットワーク利用を減少させ、ネットワーク資源の利用効率や、社員の業務効率改善をはかることが出来る。

2.1.2 Network Address Translation (NAT)

NAT は、2 つの IP アドレスを相互に変換する技術であり、1994 年に Internet Engineering Task Force (IETF) が RFC 1631 として策定した [2]。もっぱらプライベートアドレスとグローバルアドレスの変換に用いられる。また一般的に NAT と呼ばれる範囲には、Network Address Port Translation (NAPT) または IP マスカレード (IP Masquarade) と呼ばれる、IP アドレスだけでなくポート番号もともに変換する技術も含まれる。

1990 年代の初頭、インターネットに接続するノードが飛躍的に増大し、IP アドレスの枯渇が問題になっていた。この解決のために、プライベートアドレスとともに導入されたのが NAT である。

NAT は内部ネットワークと外部ネットワークを接続するゲートウェイで用いられる。

プライベートアドレスが割り振られた内部ネットワークのノードと、グローバルアドレスが割り振られたゲートウェイで、相互に IP アドレスを変換することで機能する。具体的には、ゲートウェイが通過するパケットの送信元 IP アドレスと送信先 IP アドレスを書き換えることで機能する。

IP マスカレードでは、内部ネットワークの複数のノードを、外部ネットワークのアドレス 1 つに変換することができるため、グローバルアドレスの数を超えてノードをネットワークに接続することができる。このため、接続ノード数に比べて割り当てられた IP アドレスが少ない日本では普及が進んでおり、家庭向けや中小企業向けのルータには IP マスカレードが標準搭載されることが多い。

しかし NAT を用いた場合、プライベートアドレスを用いた内部ネットワークのノードを、外部ネットワークから一意に識別することができなくなるため、外部からの接続性が失われることがほとんどである。

2.2. クライアント・サーバモデルの限界

前節では、ファイアウォールや NAT の導入によって、悪意ある通信の増加、IP アドレスの枯渇という課題の解消がはかられたことを述べた。

しかしこの副作用として、ノードが相互に接続可能という性質である “相互接続性” が失われることになった。そのため現在のインターネットの接続モデルは、ノードを、接続を受け付けサービスを提供するサーバと、接続を行いサービスを受けるクライアントの 2 種類に分類して固定化するクライアント・サーバモデルが主流となっている。

本節では、クライアント・サーバモデルについて、その特徴と問題点を述べる。

2.2.1 クライアント・サーバモデル

従来のインターネットは、伝達する情報が生じた際に、伝達する相手に対して接続を行い、情報を提供するモデルとなっていた。これは、現在は Peer-to-Peer (P2P) と呼ばれているモデルである。しかし、ファイアウォールや NAT により相互接続性が保証されなくなったため、外部ネットワークから、ファイアウォールで保護されたノードや、NAT とプライベートアドレスを利用したノードに接続することは不可能にになり、P2P モデルを維持することが困難になった。

そのため現在のインターネットは、ノードを、接続を受け付けサービスを提供するサーバと、接続を行いサービスを受けるクライアントの 2 種類に分類して固定化するクライアント・サーバモデルが主流となっている。グローバルアドレスを割り当て、ファイアウォールの設定を行うことで接続を受け付けることが可能なノードをサーバとして用意することで、ファイアウォールに保護されたノードや NAT の内部にあるノードなど、外部からの接続を受け付けることができないノードであっても、サービスを受けたい時にサーバに接続することでサービスが利用可能となる。

クライアント・サーバモデルでは、クライアントは外部から接続可能である必要はない。その代わり、サービスを受ける際は常に、自身から接続を行う必要がある。この仕組みを、情報を引き出す構図からプル (Pull) 型と言う。しかしプル型には、以下の問題がある。

- 情報が更新されたことの通知が遅れる
- 情報が更新されたことを確認するための要求の負担が生じる

プル型では、能動的に接続をしない限り新たに情報を取得することができない。そのため定期的にサーバに接続することで、情報の更新について確認し、更新されていた場合に情報を取得する。クライアントは、能動的に確認をするまで、情報が更新されているかどうかを知ることができない。そのため情報が更新されてから、その更新が伝わるまでの間に遅れが生じる。

更新確認の頻度を上げることで、この遅れの影響を小さくすることができる。しかし確認の頻度に比例してサーバの負荷が高まるため、更新確認の負荷によって本来のサービスの提供に影響を与えることが少なくない。

またインターネットで扱う情報そのものも飛躍的に増大しており、サーバやサーバが接続されたネットワークやゲートウェイなど、サービスを提供する側の負担が大きくなりつつあるのも問題である。現在は増大する負荷に対して、異なるネットワークに Web サーバを分散配置する Contents Delivery Service (CDN) など、処理を分散することで対処している。しかし処理の分散には費用がかかるため容易に用いることはできず、また情報更新の通知が遅れる問題は解決できない。

2.2.2 P2P モデル

クライアント・サーバモデルでは、情報更新の通知が遅れることと、サーバ側の負担が大きくなりつつあることが問題であると述べた。これに代わるものとして、当初のインターネットの接続モデルである Peer-to-Peer (P2P) モデルが注目されている。

P2P モデルでは当初のインターネットと同様に、ノードの役割を固定化せず、発信する情報があるノードが、その情報を伝達すべきノードに対して接続することで情報を伝達する。現在の P2P モデルでは、当初のインターネットで用いられていた P2P モデルに比べて、多数のノードから目的の情報を検索し、情報を保持するノードを動的に特定する技術を併用することが多い。

現在の P2P モデルは、接続モデルにあわせて大きく 2 種類に細分化できる。一つは、情報の検索を含めてすべて P2P で行うピュア P2P である。もう一つは、情報の検索など一部の処理にクライアント・サーバの型を取り入れ、情報の伝達のみ P2P で行うハイブリッド P2P である。

いずれの P2P モデルでも、クライアント・サーバモデルの問題であった、サーバの負担を軽減することができる。とくにピュア P2P モデルでは、ノードはクライアントにもサーバにもなり、固定したサーバが不要となるため、問題を根本的に解決することができる。しかしピュア P2P の実現には、かつてのインターネットと同様に、P2P ネットワークに参加するすべてのノードで相互接続性を確保しなければならない。一方ハイブリッド P2P では、部分的にクライアント・サーバモデルを用いているため、P2P ネットワークに参加するすべてのノードで相互接続性を確保する必要はない。しかしそのためクライアント・サーバモデルの問題を完全には解消できない。

現在は、ピュア P2P の実現が困難であり、またハイブリッド P2P におけるサーバ負荷の問題を解消するため、両者の中間となる P2P モデルも実際に用いられている。P2P ネットワークに参加するノードのうち相互接続が可能なものに、情報の検索などの機能を集中させ、ハイブリッド P2P におけるサーバと見立てることで機能する。しかしこの中間モデルでは、サーバの役割を担うノードが一定以上なければ、安定した P2P ネットワークは実現できない。これらのノードは一般のサーバと異なり常に利用可能とは限らず、一般のノードと比べても動作の安定性に大差はなく、本来のクライアント・サーバモデルにおけるサーバほど安定性を期待できない。そのため、P2P ネットワーク全体の安定性を維持するのは困難である。

このように、安定した P2P ネットワークの実現にはピュア P2P が非常に有効であり、ピュア P2P を実現するために相互接続性を確保することが重要となっている。

また、参加するノードを事前に限定するのは、ネットワーク管理者の負担となる。この場合は参加するノードの増加に比例してその負担も増大するため、P2P ネットワークをより容易に実現するには、参加するノードを事前に限定せずに済む方法である必要がある。しかし同時に、信頼に足る P2P ネットワークを構築するには、接続するノードを認証可能である必要がある。

現在のクライアント・サーバモデルを代替する P2P モデルの実現には、以下の点が必要であると考える。

- エンドノード間の相互接続性
- 参加するノードを事前に設定する必要がない
- 参加するノードは認証可能でなければならない

2.3. 認証

2.3.1 ID とパスワード

インターネットの初期から、利用者によって異なるサービスを提供するために、ID とパスワードが用いられていた。たとえば初期から存在していた電子メールでは、利用者ごとに異なるメールボックスを用いており、利用者を特定するために ID とパスワードを用いた。パスワードは当人しか知り得ない秘密の単語であるから、その単語を知っているかどうかが本人であることの確認となりうる。このように、任意の対象物と、その対象物の身元を確実に関連づけるための作業を認証と呼ぶ。ID とパスワードは利用者の認証の例であるが、対象物には人物に限らずサーバや組織なども含まれる。

現在、インターネットは社会基盤として用いられるようになり、インターネット上で提供されるサービスは増加、多様化している。銀行や証券会社の口座を参照し、取引することができるようになったほか、行政の手続きの一部もインターネットで行えるようになった。また、インターネットを用いた Business-To-Business (B2B) と呼ばれる企業間取引は、業種や企業の規模を問わず広く普及している。

提供されるサービスの増加に比例して利用者の認証を行う場面も増加している。しかし標準的に用いられているIDとパスワードでは、以下の事情から十分なセキュリティを保てないことが指摘されている。

- IDとパスワードを用いる場面の増加につれ、IDとパスワードの記憶が困難になり、結果として同一のものを使うことが多い
- 様々なサービスで同一のものを使うと、一つのサービスでIDとパスワードが流出した際に、ほかのサービスで不正利用される
- コンピュータ能力の向上により、総当たり法によるパスワードの類推が容易になりつつある

このため、IDとパスワードにかわって、公開鍵暗号方式に基づいた電子署名・電子認証の技術と、その技術を用いたPublic Key Infrastructure (PKI, 公開鍵基盤)の仕組みに注目が集まっている。

2.3.2 公開鍵暗号方式

初期から現在に至るまで、インターネット上で広く用いられている暗号化方式は共通鍵暗号方式という。初期の暗号理論を元にしており、コンピュータの誕生以前から様々な場面で用いられてきた。しかし暗号化と復号化で共通の鍵を使う共通鍵暗号方式では、ネットワークで情報をやりとりする場合、第三者に知られることなく送受信者間で安全に鍵を共有する必要がある。これは鍵配送問題と呼ばれ、共通鍵暗号方式では解決が困難な問題である。

1970年代に開発された公開鍵暗号方式[6]は、この鍵配送問題を抜本的に解決した。公開鍵暗号方式には、暗号化に用いる鍵と復号化に用いる鍵が異なるという特徴がある。この2つの鍵には数学的な関係があり、一方の鍵で暗号化した文書は、その鍵で復号化することはできず、もう一方の鍵でのみ復号化が可能という特殊な性質を持つ。このため、この一方の鍵を公開鍵として広く公開し、自身への暗号化の際に用いてもらい、もう一方の鍵を自身の公開鍵で暗号化された暗号文を復号化するために機密に保持することで、鍵配送の問題を考えずに、暗号のやりとりをすることが可能となる。

暗号化に用いる公開鍵は漏洩しても問題がなく、また復号化に用いる秘密鍵は本人のみが機密に保持するものであって配送の必要がないため、通信の段階で鍵が漏洩する可

能性は理論上まったくない。そのため第三者に鍵を不正に利用されたり、暗号が解読される危険性は、共通鍵暗号方式に比べて遙かに低く、セキュリティを高く保つことが可能である。

また、秘密鍵を共通鍵暗号を用いて暗号化することもよく行われる。この場合、秘密鍵を利用するには共通鍵暗号で秘密鍵を復号化しなければならない。これを鍵の活性化 (activate) と呼ぶ。第三者が不正な方法で、活性化されていない、非活性の秘密鍵だけ手に入れた場合でも、活性化のための共通鍵暗号が分からなければ鍵は利用できない。同様に活性化のための共通鍵暗号だけ手に入れた場合でも、対象となる秘密鍵が分からなければ鍵は利用できない。秘密鍵自身とその鍵を活性化するための共通鍵暗号の双方がなければ秘密鍵を利用できないため、不正に利用される危険性を大きく低減することができる。

公開鍵暗号方式は暗号だけでなく電子署名にも用いられる。電子署名とは、受信者のもとに届いた文書が送信者の意図した通りのものであることを証明し、第三者による改ざん、なりすましが行われていないことを確認するための技術である。

前述のとおり、公開鍵と秘密鍵には数学的な対応があり、一方の鍵で暗号化した文書は、もう一方の鍵で復号化が可能という性質を持つ。そのため送信者は暗号化とは逆に、送信者の秘密鍵を用いて文書を暗号化すると、受信者は送信者の公開鍵を用いて復号化することが可能である。

秘密鍵を正しく利用できるのは秘密鍵の所有者だけであるため、復号化が正しく行えた場合、文書は秘密鍵の所有者によって暗号化され、改ざんが行われていないということが確認できる。また、このことから本人であることを認証する手段としても用いることができる。

第3章　既存技術

本章では、前章で概説したインターネットに要求されている事項を実現するために提案されている既存技術について述べる。

3.1. 仮想ネットワーク技術

仮想ネットワーク技術とは、物理的なネットワークの上に仮想的なネットワークを構築するための技術である。物理的なネットワークに設置されたファイアウォールの、特定のポートを通過する設定にすることで、物理的な制限に影響を受けず、自由な形でネットワークを構築するために用いられる。このため相互接続性の確保が容易であり、P2Pモデルを実現する方法の一つと考えられる。

仮想ネットワーク技術は、仮想ネットワークを実現するための OSI 参照モデルのレイヤーによって性質が異なる。アプリケーション層で行うものとして SOCKS、ネットワーク層で行うものとして IP Virutal Private Network(VPN)、またデータリンク層で行うものの代表例として SoftEther について説明する。

3.1.1 SOCKS

ファイアウォールの保護下にある内部ネットワークのノードと、外部ネットワークのノードで相互に通信するための方法として SOCKS がある。

SOCKS では、外部ネットワークと内部ネットワークの境界に、ファイアウォールに保護されないノードを設置し、外部と内部の接続のための代理サーバとする。代理サーバは SOCKS サーバとよばれる。SOCKS サーバは、外部と内部のそれぞれから TCP/IP による接続を受け付ける。

ファイアウォールの保護下にあるノードがファイアウォールの外部にあるノードに、またファイアウォールの外部にあるノードがファイアウォールの保護下にあるノードに接続を希望する場合、SOCKS に対応したアプリケーションは、まず SOCKS サーバに接

続を行い、接続を希望する相手先ノードを通知する。SOCKS サーバは、必要に応じて接続を希望するノードやエンドユーザを認証し、希望する相手先ノードに TCP/IP によって接続する。

この方法は、アプリケーションが対応することによって、利用者には相互に接続可能であるように見える。実際には、通信のすべての内容を SOCKS サーバが一度受信し、その後、目的のノードに再送信することで機能する。

SOCKS では、通信がすべて SOCKS サーバを介すため、ネットワーク管理者が通信を捕捉しやすいという特徴がある。しかし、SOCKS の利用にはアプリケーションの対応が必要である。また SOCKS サーバが一度通信を受信して再送信するため、SOCKS サーバの負担が大きく、スループットは大きく低下するという問題がある。

3.1.2 IP VPN

IP VPN は、OSI 参照モデルにおけるネットワーク層で仮想ネットワークを構築するための技術である。仮想ネットワークの IP パケットをカプセル化し、暗号化を施した上で、物理ネットワークの IP パケットのデータ部として送信することで仮想ネットワークを構築する。

特に断りなく IP VPN と呼ぶ場合は、IPsec の暗号化と認証の仕組みを用いたものを指すことが多い。IPsec には、IP パケットのデータ部分であるペイロードを暗号化する機能や、IP パケットの内容が途中で改ざんされていないことを保証する機能がある。これらの片方または両方を用いることで、仮想ネットワーク上の通信路を安全に保つことができる。

これによって、物理的に遠地のネットワークにあるノード同士を、同一のセグメントにあるように見せかけることができる。IP VPN は仮想ネットワーク技術の中で最もよく利用されており、対応する OS やネットワーク機器は非常に多い。

しかし IP VPN は主に、企業内や組織内などの信頼されたノードによる、閉じた仮想ネットワーク構築の技術であり、IP VPN で構築された仮想ネットワーク上にファイアウォールを設置することはほとんどない。そのため悪意ある第三者が、物理的なネットワークのノードの制御を奪った場合、そのノードが参加している仮想ネットワークを経由し、他のファイアウォールの保護下にあるノードの制御まで奪われる危険性がある。また、経路を暗号化することによって通信の秘密を保護するが、このことから物理的なネッ

トワーク管理者が通信内容を把握できないため、悪意ある第三者による攻撃を防止するどころか、発見することもできない。

このような理由から、IP VPN がまったく利用できないか、あるいは管理者の完全な把握のもと、特定のノード間でのみの利用しかできないように制限されていることが少なくない。

3.1.3 SoftEther

SoftEther は、OSI 参照モデルにおけるデータリンク層で仮想ネットワークを構築するための技術である。Windows 上で動作する仮想的な Ethernet のネットワーク・インタフェース・カード (NIC) と、Ethernet のスイッチング・ハブの組み合わせで機能する。

データリンク層において仮想ネットワークを構築するため、ネットワーク層を IP に限定せず、他のネットワークプロトコルも利用可能である点が特徴である。

ネットワーク層とデータリンク層という違いはあるが、仮想ネットワーク上通信が暗号化されるためネットワーク管理者の捕捉を受けにくいという点など、IP VPN と同様の欠点を持つ。

3.2. PKI に関する技術

PKI とは、電子認証・電子署名の技術と、公開鍵が正当なものであることを保証するための機関や、その機関の運営ポリシーなどの技術以外の仕組みを含めた総称である。本節では、PKI について概要を説明し、PKI を実現するための規格と技術について述べる。

3.2.1 PKI

前章で述べた、公開鍵暗号方式に基づいた電子認証・電子署名の技術を用いることで、共通鍵暗号方式で問題となる鍵配送問題を解消し、暗号に用いる鍵を第三者に不正に用いられる危険性を低減することができる。しかし、この技術だけでは不十分である。

暗号の際には、受信者の公開鍵を用いて暗号化を行うが、この時に用いる公開鍵は、確かに受信者のものでなければならない。悪意ある第三者が暗号文を傍受するために、自身の公開鍵を正当な受信者の公開鍵であると偽装して送信者に配布したとする。送信者がその公開鍵を用いて暗号化すると、悪意ある第三者に傍受された際に解読されてしま

う。そのため、公開鍵が確かに受信者のものであるということを保証する仕組みが必要である。

この保証に用いられる基盤を PKI という。PKI とは、電子認証・電子署名の技術と、公開鍵が正当なものであることを保証するための機関、その機関の運営ポリシーなどの技術以外の仕組みをすべて含めた総称である。

PKI では、信頼の拠点となる認証局 (Certification Authority: CA) を設置し、CA が公開鍵に電子署名することで公開鍵の正当性を保証する。CA によって正当であると保証され、電子署名された公開鍵は、公開鍵証明書、あるいは単に証明書と呼ばれる。CA は、場合によっては他の CA によって正当性が保証される場合がある。公開鍵の利用者は、公開鍵を署名したとされる CA が正当なものであると信頼できるなら、公開鍵も正当なものであると判断できる。CA が信頼できるかどうかの判断は、最終的には利用者にゆだねられる。PKI は一つの CA を信頼することで、その CA が署名した公開鍵についても信頼する、という信頼の連鎖によって機能する。

PKI は暗号化、電子署名など様々な用途に用いることができる。その中でも証明書による電子認証は、ID とパスワードを置き換える手段として注目を集めている。現在の ID とパスワードは、本人しか知り得ない単語を入力させることで本人であることの証明としているが、短い単語であることから漏洩などの問題があると前述した。証明書を用いることで、これらの問題の多くは解消できる。

証明書を用いることで、たとえば銀行や証券会社の口座操作などの重要な取引がより安全にできるほか、本人確認を必要とする行政手続きなどが、インターネット上で容易に行えるようになった。今も PKI の普及は進んでおり、今後も証明書を用いた電子認証は広く普及し、金銭を伴う取引や行政手続きの多くがインターネット上で行えるようになると考える。

3.2.2 X.509 と PKCS

PKI そのものは規格や仕様を指すものではないため、実際の運用では、証明書をやりとりするにあたって規格が必要となる。

証明書をやりとりするための標準型式としては、1998 年に ITU-T が勧告した X.509 が広く用いられている [7]。1999 年には IETF が、インターネットで利用することを目的として X.509 を RFC とした [8]。

X.509 は、インターネットにおける証明書の事実上の標準であり、多くのアプリケーションで用いることができる。UNIX 系 OS などの Secure Shell (SSH) の実装である OpenSSH は OpenSSL[9] ライブラリを用いているが、OpenSSL でも標準の証明書の型式として利用している。

X.509 の証明書には表 3.1 の内容が含まれる。拡張項目については、重要な項目のみ記した。

表 3.1 X.509 形式の証明書に含まれる内容

項目名	項目の説明
基本的な項目	
バージョン番号	証明書型式のバージョン 現在は X.509 バージョン 3 を用いることがほとんどである
証明書のシリアル番号	証明書の発行者において証明書を一意に識別するための数 発行者ごとに一意に識別可能ならば、どのような数であってもよい 発行の日付を用いたり、企業では社員番号などを用いることもある
暗号に関する情報	電子署名に用いられる暗号アルゴリズム、ハッシュ関数の情報
証明書の発行者	証明書の発行者の名称
有効期限	証明書を利用可能な有効期限の開始日と終了日
公開鍵の所有者	証明書によって証明される人物の名称と、 その人物の会員番号、電子メールアドレスといった人物固有の情報
公開鍵の本体	公開鍵の本体
バージョン 3 で拡張された項目	
鍵利用目的	証明書に記載されている公開鍵を用いることが可能な利用目的
CRL 配布ポイント	証明書失効リストを配布する URI

また、証明書の符号化の方式として、米 RSA Data Security 社 (現 RSA Security 社) [10] が策定した公開鍵符号化の規格である Public Key Cryptography Standards (PKCS)[11] が広く用いられている。PKCS は複数の規格が集合した名前であり、証明書の形式の他にも、秘密鍵や暗号の仕様や、秘密鍵と証明書を単一のファイルに保持するための型式など、PKI を適用する上で必要な様々な規格や型式を定めている。

3.2.3 S/MIME

PKI の利用例として、米 RSA Data Security 社が提案した Secure MIME (S/MIME) がある。S/MIME は、インターネット上の電子メールで用いられる代表的な符号化方式である。電子メールの暗号化と、署名を行うことができる。1998 年に IETF によって RFC となった。

もともとインターネットの電子メールは ASCII 文字のみを扱っていた。しかし、電子メールの普及に従って、漢字やひらがなといったアジア圏のマルチバイト文字や画像、音声、動画などを扱うための Multipurpose Internet Mail Extension (MIME) の仕様が追加された。この仕様を利用して、暗号化や電子署名を施したメールを扱えるようにした仕様が S/MIME である。

証明書や電子署名の扱いには PKCS を用いる。暗号には PKCS #1 “RSA Encryption” を、メッセージの書式には PKCS #7 “Cryptographic Message Syntax” などを用いる。また S/MIME の書式は MIME を用いているため、S/MIME に対応していない電子メールソフトウェアであっても、MIME に対応した電子メールソフトウェアであれば、受信したメールによって誤動作するといった深刻な影響はない。

暗号化には、公開鍵暗号として RSA を、また併用する秘密鍵暗号として、RC2、DES を用いる。電子署名には、ハッシュ関数として SHA-1 または MD5 を用いる。

S/MIME による暗号化および電子署名は、以下のような流れになる。

- 証明書の準備
  1. CA が利用者のために、公開鍵と秘密鍵の鍵対を生成する
  2. CA は CA 自身の秘密鍵によって、この利用者の公開鍵に対して電子署名し、証明書を生成する
  3. CA は利用者の証明書と秘密鍵を、利用者に配布する
- 暗号の利用
  1. 送信者は暗号文を送りたい相手の証明書を入手する
  2. 送信者は、入手した証明書が CA に正しく署名されたものかを検証する
  3. 送信者は CA を信頼するかどうかの判断を下す
  4. 送信者は、受信者の証明書に記載されている公開鍵を用いて公開鍵暗号化を施す

5. 送信者は暗号文を送信する
6. 受信者は受信した暗号文を、自身の秘密鍵を用いて復号化する

- 電子署名の利用

1. 送信者は、自身の秘密鍵を用いて文書に公開鍵暗号化を施す
2. 送信者は署名した文書に、署名に用いた証明書を添付して送信する
3. 受信者は受信した、署名された文書から証明書を取り出す
4. 受信者は、取り出した証明書が CA に正しく署名されたものかを検証する
5. 受信者は CA を信頼するかどうかの判断を下す
6. 受信者は、署名された文書を、添付された証明書を用いて検証する

3.3. 既存技術のまとめと問題点

本章で挙げた技術によって、ネットワーク上に仮想的に新たなネットワークを構築し、それによって相互接続性を維持し、P2P を実現可能であるが、それぞれの技術には問題点があると述べた。それらの問題点を表 3.2 にまとめた。

表 3.2 既存技術の比較

	SOCKS	IP VPN	SoftEther
アプリケーションの対応	×必要	○不要	○不要
スループットは低下しないか	×大きく低下	×低下	×低下
ネットワーク管理者が把握できるか	○できる	×できない	×できない
認証の種別	ID/パスワード	ID/パスワード	ID/パスワード

仮想ネットワーク技術は、接続する対象をあらかじめ限定した環境では有効に機能する。しかし、P2P ネットワークに接続するノードを増減させる際には事前の設定が必要であるため、運用の柔軟性に欠ける。このため、これらの技術の利用を前提とした P2P アプリケーションは実現に至っていない。

今後、アドレス数の拡張された IPv6 が導入されることによって、NAT で失われた相互接続性は回復することができる。しかし今後も悪意ある通信が減少する可能性は少な

いため、ファイアウォールは存在し続けると考える。したがって、P2P アプリケーションの容易な実現にはファイアウォールと共存可能な技術が必要である。

第4章　提案

本章では、前述の問題を解決するために、エンドノードの主導によってファイアウォールを動的に再構成するファイアウォール制御技術について提案する。また、ファイアウォールを制御し、通信を可能とする上で必要なネットワークやノードの安全性を維持するための方法として、エンドユーザに立脚した、エンドユーザ同士による信頼モデルについて提案する。

4.1. エンドノードによるファイアウォール制御

ファイアウォールは、外部のネットワークから内部のネットワークを保護するという性質上、慎重に制御される。具体的には、既知の危険だけでなく、将来発生する可能性のある未知の危険にも効果を発揮することが望まれるため、多くの接続を遮断するよう設定されている。またそれぞれのエンドノードについて個別にファイアウォールの設定をすることは、設定が膨大になることや、その設定にかかる人的費用の点から困難であり、保護下にあるすべてのエンドノードについて一律の保護を行うことがほとんどである。このため最も弱点であるエンドノードを保護するために、すべてのエンドノードについて、より慎重で安全性の高い設定となっている。

しかし安全性と利便性は相反する関係にあるため、安全性の代償として、接続に関する柔軟性は大きく失われた。現在では、ファイアウォールに保護されたノードは、一律に相互接続性が失われているといってよい。しかし、何らかのポリシーに基づき安全と認定されたノードに関しては、ファイアウォールの保護は最小限に留めることが可能であるといえる。それによって、相互接続性を確保することができる。エンドノードの安全性に基づいて個別にファイアウォールを設定する機構があれば、適切にファイアウォールの保護を行い、かつ相互接続性を確保することが可能であると考える。

そのため、ファイアウォールの制御の主導を、エンドノードおよびエンドユーザに委譲

する機構について提案する。本機構は、エンドノードとファイアウォールが連携し、ファイアウォールがエンドノードの要求に従い、またエンドノードの設定や状態およびエンドユーザの能力に応じて、ファイアウォール自身を再構成することを可能とする。また、接続を要求する側と接続を受け入れる側の、双方のファイアウォールが相互に連携することでエンドノード間の相互接続を可能とする。

接続の可否の主体がエンドノードになり、エンドノードがファイアウォールの制御を主導することで、エンドノード、エンドユーザに基づいた保護の設定が可能となり、相互接続性を確保できる。

本研究では、外部ネットワークからの悪意ある通信に耐えることが可能で、かつ外部ネットワークに対して悪意ある通信行う可能性も小さいと判断できることが可能なノードを、安全なノードと定義する。具体的には、以下の要件を満たすノードを安全とみなす。

- 不要なソフトウェアを動作させていない
- 必要なソフトウェアは、最新版にアップデートされている
- コンピュータウイルスをチェックするソフトウェアが正常に動作している
- ノードの利用者が特定できている

これらの安全なノードに関しては、ファイアウォールの保護は最小限に抑えることが可能であると考える。

4.2. エンドユーザ間の信頼の確立

前述のように、ファイアウォールは外部のネットワークから内部のネットワークを保護するためにある。悪意ある通信はファイアウォールで遮断しなければならないが、エンドノードによるファイアウォールの制御にあたって、悪意ある通信の遮断の能力が衰えてはならない。悪意ある第三者がファイアウォールの保護下にあるエンドノードをだまして、通信を許可させることがないように、許可の判断は慎重に行わなければならない。判断を行うには、接続を行う相手を正しく認証する方法が必要である。

また、エンドノード間で接続を確立するには、接続を要求するエンドノードの側にあるファイアウォールだけでなく、接続を受け付けるエンドノードの側にあるファイアウォールの制御も必要となる。しかし、接続を受け付ける側のファイアウォールは、自分の保護下ではないンドノードに関しては情報を保持しない場合がほとんどであるため、接続を要求するエンドノードを信頼することはきわめて困難である。

そこでエンドノード主導によるファイアウォール制御のため、通信を行うエンドノードとエンドユーザを判断、認証するための信頼モデルを提案する。

この信頼モデルは、接続を行う側、接続を受け付ける側の双方のエンドノードの利用者、つまりエンドユーザによる信頼関係を基本とする。

通信においては、エンドノードは一般に意志を持たないが、エンドユーザの発意に基づき実際に行為を行うため、この両者の区別は特に行わない。しかし、エンドノードの認証とエンドユーザの認証は異なるため、認証においてはこの両者は区別される。従って、通信に関する記述に関しては両者を区別せずエンドノードとして統一するが、認証に関する記述においては両者を区別し、個別に表記する。

本提案では、エンドユーザ間の信頼に PKI を用いる。接続を行う側が、PKI による電子証明書によって、自分の実在性および、所属、権限、能力といった属性を、接続を行う相手に提供する。接続を受け入れる側は、これらのエンドユーザの情報を元に接続の可否を判断することができる。

エンドユーザと、エンドユーザが利用しているエンドノードの関連付けは、エンドノードへのログイン作業によって行う。したがってこの関連付けが正当なものであるかどうかは、OS のエンドユーザの認証が正常に行われているかどうかに依存する。しかし現在はほとんどの OS でエンドノードへのログイン作業が必要であるため、本提案ではエンドノードとエンドユーザの関連付けについては、行われているものと仮定する。

本提案では、ファイアウォールがエンドノードを保護下に置くために認証する手段については定義しない。エンドノードは、DHCP による MAC アドレス認証や、認証 VLAN などの仕組みによって認証され、ファイアウォールの保護下におかれる。ファイアウォールの保護下に置かれているエンドノードは、ファイアウォールが保護を開始するにあたって、何らかの方法で認証したものとみなす。

ここでファイアウォールにとっては、自分が保護下に置くエンドノードと、そのエンドノードの利用者であるエンドユーザを信頼できるのであれば、そのエンドユーザが信

頼する者も信頼することができるはずである。したがって、接続を行う側、受け付ける側の双方にあるファイアウォールが、各々のエンドノードとエンドユーザを認証し、またエンドユーザ間においても信頼が成り立つのであれば、ファイアウォールは自分の保護下にあるエンドノードへの信頼によって、自分の保護下ではないエンドノードの情報を知らずとも、接続を許可することができる。

図 4.1 で、これらの信頼の関係について図示する。図中の (A)〜(E) の矢印は、信頼関係について示している。 (A) が、エンドユーザ相互による信頼関係である。(B) は、エンドユーザがエンドノードにログインすることで成り立つ信頼関係、(C) はファイアウォールがエンドノードを認証することによって成り立つ信頼関係を示す。(D) はファイアウォール間の信頼関係を示す。(E) は、エンドユーザとファイアウォールによる認証局への信頼を示す。

本提案では、(A) のエンドユーザ相互による信頼関係に基づいて、ファイアウォールを制御する。エンドユーザ相互の信頼関係を成り立たせるには、信頼関係を構築する手段が必要となる。そのため本提案では、(B)、(C)、(D) という、既存の信頼関係の構築の手段を用いて信頼関係を構築し、その信頼関係を連鎖することで、エンドユーザ相互の信頼関係を成立させる。

この信頼関係によって、より安全なファイアウォールの制御が可能となる。

本提案によるファイアウォール制御のための機構と、制御のための信頼モデルによって、ファイアウォールによる保護を受けつつ相互接続性を確保することが可能である。これによって、現在のクライアント・サーバモデルを代替可能な、広域で信頼のおける P2P ネットワークの実現が容易になると考える。

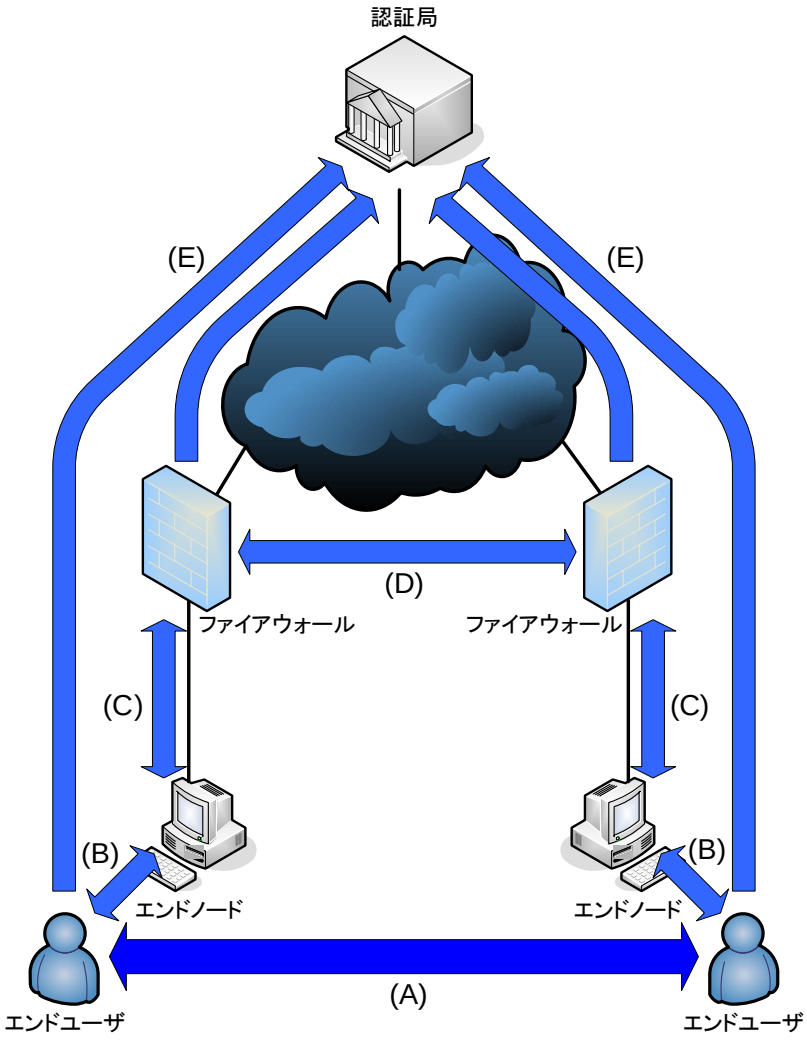
認証局
(E)
(E)
(D)
ファイアウォール
ファイアウォール
(C)
(C)
(B)
(B)
エンドノード
エンドノード
(A)
エンドユーザ
エンドユーザ

図 4.1 信頼の関係

第5章　設計

本章では、前章の提案に基づいたファイアウォール制御機構の設計について述べる。まず全体の概要を述べ、続いてそれぞれの構成要素、および処理の流れについて概観を述べる。

5.1. 設計の概要

エンドノード間の相互接続性の回復のため、エンドノードとファイアウォール間、およびファイアウォール相互間を連携させる機構 “Firewall End-node Intercommunicating Traverser”(FEIT) を設計した。本提案機構は、エンドノード側に導入するシステムと、ファイアウォール側に導入するシステムの 2 種類から成り立っている。ここでは前者をエンドノードシステムと呼び、後者をゲートウェイシステムと呼ぶ。本提案機構を利用するには、エンドノードにエンドノードシステム、ファイアウォールにゲートウェイシステムが導入されていなければならない。

これらのシステムの役割について説明する前に、以降の説明で用いる単語について定義する。

自エンドノード

任意のファイアウォールから見て、自分の保護下にあるエンドノードのこと。また、任意のエンドノードから見て、自分自身のこと。

自ファイアウォール

任意のファイアウォールから見て、自分自身のこと。また、任意のエンドノードから見て、自分が所属するネットワークに存在し、自分を保護下におくファイアウォールのこと。

他エンドノード

任意のファイアウォールから見て、自分の保護下にないネットワークにあるエンド

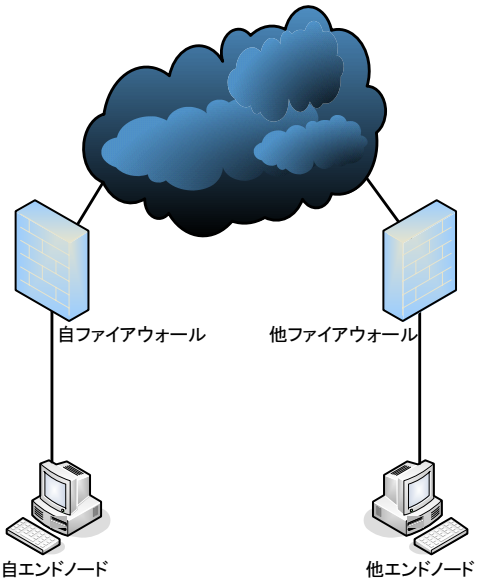

図 5.1 自他ファイアウォール、エンドノードの関係

ノードのこと。また任意のエンドノードから見て、自分の所属ではないネットワークにあるエンドノードのこと。

他ファイアウォール

任意のファイアウォールから見て、自分の保護下にないネットワークにあるファイアウォールのこと。また任意のエンドノードから見て、自分の所属ではないネットワークにあるファイアウォールのこと。

自他という接頭語を含む単語はすべて、ある一つの立場に立脚した相対的なものである。したがって「他ファイアウォール」や「他エンドノード」に立脚した場合、自他はすべて入れ替わる。

以上の関係を図 5.1 に示す。

以下に、エンドノードシステムとゲートウェイシステムが持つ役割をあげる。

- エンドノードシステム
  - 自エンドノードとエンドユーザの認証を要求する

- 自ファイアウォールに、他エンドノードへの接続要求を行う
- 自ファイアウォールから転送された、他エンドノードからの接続要求について、可否を判断する

- ゲートウェイシステム
  - 自エンドノードとエンドユーザを認証する
  - 自エンドノードから他エンドノードへの接続要求を受け付ける
  - 自エンドノードから他エンドノードへの接続要求を、他ファイアウォールへ転送する
  - 自エンドノードの要求によって、自ファイアウォールの設定を変更する
  - 他エンドノードから自エンドノードへの接続要求を、自エンドノードへ転送する

エンドノードシステム、ゲートウェイシステムについては 5.2 節で詳しく述べる。また、エンドノードシステムとゲートウェイシステム、またゲートウェイシステム相互間の連携による処理の流れについては、5.4 節で述べる。

5.2. 提案システムの構成要素

5.2.1 エンドノードシステム

ゲートウェイシステムの導入されたファイアウォールの保護下にあるエンドノードに導入されるのがエンドノードシステムである。エンドノードシステムの持つ役割は、エンドノードとエンドユーザの認証を要求し、アプリケーションの接続要求に基づいて、ファイアウォールを制御するためにゲートウェイシステムへ接続の要求をすること、また他のエンドノードからの接続要求の可否を判断することである。

A. 自エンドノードとエンドユーザの認証を要求する

エンドノードシステムはエンドユーザによって任意のタイミングで起動される。エンドノードシステムは起動直後に、自ファイアウォールのゲートウェイシステムに対して、自エンドノードとエンドユーザについて認証するよう要求を行う。

この認証要求には、自エンドノードの IP アドレスと、エンドユーザをグローバルユニークに識別するための識別子が含まれる。今回の設計では識別子として、個人を識別するにあたって最も簡単で、かつグローバルユニークであることが保証されている E-mail アドレスを用いた。

この認証要求を、エンドユーザの秘密鍵を用いて電子署名し、自ファイアウォールのゲートウェイシステムに送信する。

B. 自ファイアウォールに、他エンドノードへの接続要求を行う

エンドノードシステムは認証完了後、提案機構に対応するアプリケーションから接続要求が行われるまで待機する。対応するアプリケーションは、TCP/IP の接続要求を行う前に、エンドノードシステムに対して接続要求を通知する。エンドノードシステムはアプリケーションからの要求をうけ、アプリケーションを待機させ、自ファイアウォールに対して接続要求を行う。

すでに自エンドノードおよびエンドユーザの認証を経た後であるため、接続要求では接続したい他エンドノードの IP アドレスと、TCP または UDP の区別、およびポート番号の 3 種類の情報を通知する。他エンドノードを保護下におく他ファイアウォールの検知はゲートウェイシステムに任される。

この接続要求は、自ファイアウォールによって、他ファイアウォールを経由し他エンドノードに伝えられる。他エンドノードが、この接続要求について許否を判断した結果は、他ファイアウォール、自ファイアウォールを経由して返答される。この返答の内容が許可であれば、返答が行われた段階で、すでに自ファイアウォール、他ファイアウォールともにファイアウォールの動的な構成を終了しているため、他エンドノードへの接続が可能である。

従って、エンドノードシステムは、停止させていたアプリケーションに、接続が許可されたことを伝達し、処理を再開させる。この後、アプリケーションは通常の TCP/IP 接続手順によって他エンドノードへ接続を確立することが可能となる。

C. 他エンドノードからの接続要求について、可否を判断する

エンドノードシステムは、自エンドノードに対する他エンドノードから接続要求を受け入れるかどうかの判断を行う。

他エンドノードからの接続要求には、他エンドノードが他ファイアウォールへ認証要求を行った際の電子署名に証明書が添付されている。エンドノードシステムはこの証明書を用いて他エンドノードのエンドユーザの情報を勘案し、他エンドノードの接続要求について可否の判断を下す。

実際の判断は、あらかじめエンドユーザが、接続要求に添付された証明書において利用する項目と、その項目の情報についての条件文を記しておいたものを元に、エンドノードシステムが自動的に判断する。たとえば、証明書の発行者が A であれば接続を許可する、とあらかじめエンドユーザが設定することで、その後受け付ける接続要求についてはエンドノードシステムが自動的に証明書を検証し、発行者が A であるかどうかを確認し、許否を判断する。

5.2.2 ゲートウェイシステム

ファイアウォールの稼働しているノードに導入されるのがゲートウェイシステムである。ゲートウェイシステムの持つ役割は、自エンドノードとエンドユーザを認証し、エンドノードシステムからの要求に従い、他のエンドノードシステムへ要求を転送することである。

D. 自エンドノードと、エンドユーザを認証する

ゲートウェイシステムは、自エンドノードのエンドノードシステムからの要求に従って、自エンドノードとエンドユーザを認証する。自エンドノードのエンドノードシステムがどのようなタイミングで認証を要求するかについては規定しない。自エンドノードおよびエンドユーザが要求した時点の情報を用い、ゲートウェイシステムは認証を行う。

自エンドノードの認証は、IP アドレスによって行う。ゲートウェイシステムの導入されたノードが自エンドノードを、ファイアウォールとして保護するノードであると判断した場合はエンドノードの認証をしたものとみなす。つまり、エンドノードの認証はファイアウォールの性質に依存する。

エンドユーザの認証には、PKI による証明書を用いる。エンドノードシステムは認証の際に、エンドユーザの情報と時刻を記した認証要求の文書にエンドユーザの電子署名を行う。ゲートウェイシステムは認証要求の文書の署名を検証し、検証に成功した上で、

証明書の記載と認証要求に記されたエンドユーザの情報が一致する場合、認証をする。この情報には、前述のとおり E-mail アドレスを用いる。

自エンドノードとエンドユーザの認証は、以下のいずれかの状態になるまで継続する。

- 自エンドノードとの接続が切断される
- エンドユーザが明示的に認証情報を消去を要求する
- 認証要求に記された有効期限に至る
- ゲートウェイシステムの管理者が設定した有効期限に至る

また、認証を継続する間、ゲートウェイシステムは自エンドノードとエンドユーザの情報を保存する。

E. 自エンドノードから他エンドノードへの接続要求を受け付ける

ゲートウェイシステムは、認証の済んだ自エンドノードから、他エンドノードへの接続要求を受け付ける。他エンドノードへの接続をゲートウェイシステムの導入された自ファイアウォールが遮断している場合は、まず自ファイアウォールのゲートウェイシステムがその接続要求を許可できるかどうか判断する。許可できる場合、接続要求にある他エンドノードを保護するファイアウォールを発見し、転送する。

ゲートウェイシステムの導入された自ファイアウォールが遮断していない場合、自身での判断は行わず、接続要求を転送する動作のみ行う。

F. 他エンドノードから自エンドノードへの接続要求を受け付ける

ゲートウェイシステムは、自エンドノードが許可した他エンドノードからの接続要求について、許否の判断を行う。

接続要求の許否の判断は、事前にゲートウェイシステムの管理者が設定した指標に沿うかどうかによって行う。認証した自エンドノードが許可しているため、自エンドノードを信頼し、接続を要求する他エンドノードについての信頼は考慮しない。しかしゲートウェイシステムの管理者は自由な判断で、許可しない接続を設定することができる。

G. 自エンドノードの要求によって、自ファイアウォールの設定を変更する

ゲートウェイシステムは、以下の状況で、接続が可能なように自ファイアウォールの設定を変更する。

- 自エンドノードからの他エンドノードへの接続要求を許可する場合
- 他エンドノードから自エンドノードへの接続要求があって、自エンドノードが接続を許可し、また自ファイアウォールも接続を許可する場合

具体的には、自エンドノードからの他エンドノードへの接続要求の場合、接続要求のある自エンドノードから、他エンドノードの、要求されたプロトコルとポート番号についてファイアウォールを設定する。他エンドノードから自エンドノードへの接続要求の場合も同様に、接続要求のある他エンドノードから、自エンドノードの、要求されたプロトコルとポート番号についてファイアウォールを設定する。

H. 自エンドノードへの接続要求を、自エンドノードへ転送する

ゲートウェイシステムは、他エンドノードからの自エンドノードへの接続要求を、自エンドノードへ転送する。

ゲートウェイシステムは、他ファイアウォールから接続要求が転送されてきた場合、認証を行った自エンドノードの一覧と照合し、適合するエンドノードが存在した場合、転送を行う。エンドノードシステムは、接続要求の受け付け用として特定のポートで接続を待ち受けており、ゲートウェイシステムが保持する認証済み自エンドノードの情報には、そのポートの番号が含まれている。ゲートウェイシステムは、自エンドノードの一覧と照合した際に、接続要求の受け付け用のポートを取得し、そのポートに接続を行って接続要求を転送する。

5.3. プロトコルと通信メッセージ

本節では、ゲートウェイシステムとエンドノードシステム間、またゲートウェイシステム相互間で認証および接続の要求を行う際のプロトコルと、そのプロトコルで利用される通信形式について述べる。

5.3.1 プロトコル

ゲートウェイシステムとエンドノードシステム間で情報をやりとりするためのコマンドとして、HELLO、CONNECT、REQUEST、INFO、QUIT の 5 種類を定義する。

HELLO

HELLO は、自エンドノードのエンドノードシステムが自ファイアウォールのゲートウェイシステムに対して制御用の接続を確立した後で、認証を要求する際に用いるコマンドである。5.2.1 項の A に対応する。

CONNECT

CONNECT は、自エンドノードのエンドノードシステムが自ファイアウォールのゲートウェイシステムに対して、他エンドノードへの接続を要求する際に用いるコマンドである。5.2.1 項の B に対応する。

REQUEST

REQUEST は、ゲートウェイシステム相互間で接続要求を転送するためのコマンドである。5.2.2 項の E の後半に対応する。エンドノードシステムからゲートウェイシステムへの接続要求のコマンドである HELLO コマンドと類似しているが、HELLO と異なり、認証要求と接続要求を一度に行うため、別のコマンドとして定義した。

INFO

INFO は、ゲートウェイシステムに関する情報を取得するためのコマンドである。INFO コマンドは情報を取得するだけであり、処理の流れには影響を受けず、また影響を与えない。取得したい情報の種別によっては、HELLO コマンドによる認証後でなければならない。

QUIT

QUIT は、自エンドノードのエンドノードシステムが自ファイアウォールのゲートウェイシステムに対して、HELLO コマンドによって行なわれた認証を解除し、制御用の接続を切断するよう要求するコマンドである。

5.4. 提案システムの処理の流れ

本章で設計した提案機構の処理の流れについて、認証要求を図 5.2 で示し、接続要求を図 5.3 で示した。

接続要求の図では、エンドノード A が接続を要求する側、エンドノード B が接続を受け付ける側である。それぞれの数字と、5.2.1 項と 5.2.2 項で説明した機能の対応は、次の通りである。5.2.1 項と 5.2.2 項における説明では、図中のエンドノード A は自エンドノードに、エンドノード B は他エンドノードに読み替える。

(1) 5.2.1 項の A に対応する。自エンドノードによる、自エンドノードとエンドユーザの認証要求である

(2) 5.2.2 項の D に対応する。自ファイアウォールによる、自エンドノードとエンドユーザの認証である

(3) 5.2.1 項の B に対応する。自エンドノードによる、他エンドノードへの接続要求である

(4) 5.2.2 項の E に対応する。自ファイアウォールによる、自エンドノードからの接続要求の受け付けと、その許否の判断である

(5) 5.2.2 項の G に対応する。自ファイアウォールによる自ファイアウォールの動的な再構成である（図中は、ファイアウォールを FW と略した）

(6) 5.2.2 項の E の後半に対応する。自ファイアウォールによる、他ファイアウォールへの接続要求の転送である

(7) 5.2.2 項の H に対応する。他ファイアウォールによる、他エンドノードへの接続要求の転送である

(8) 5.2.1 項の C に対応する。他エンドノードによる、自エンドノードからの接続要求についての、許否の判断である

(9) (8) の応答である

(10) 5.2.2 項の F に対応する。他ファイアウォールによる、自エンドノードからの接続要求についての、許否の判断である

(11) 5.2.2 項の G に対応する。他ファイアウォールによる他ファイアウォールの動的な再構成である

(12) 、(13) いずれも (10) の応答である

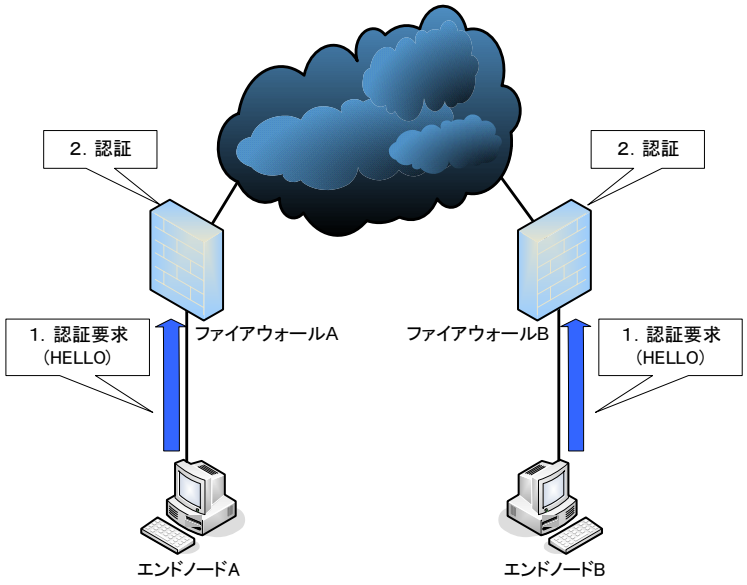

図 5.2 認証要求の処理の流れ

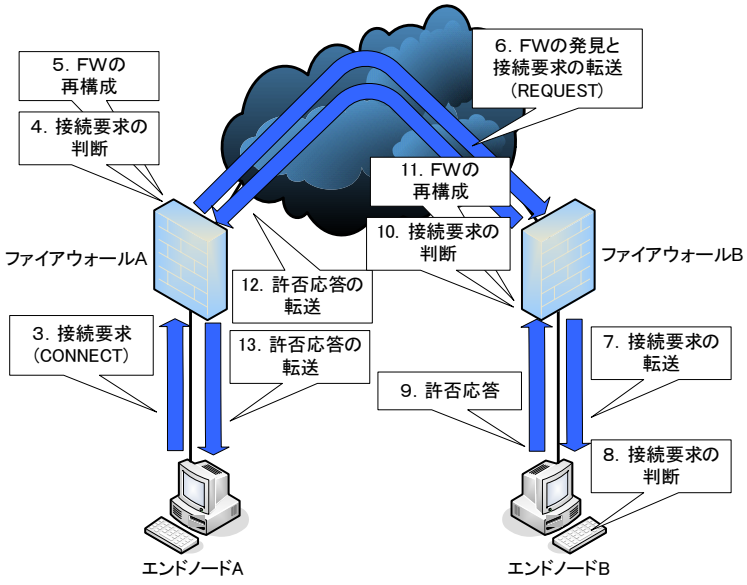

図 5.3 接続要求の処理の流れ

第6章　実装

本章では前章の設計に基づいて行った実装について述べる。まず、ゲートウェイシステムとエンドノードシステムにおけるプロトコルについて説明し、ゲートウェイシステム、エンドノードのそれぞれの実装について説明する。

提案機構の設計は IPv4/IPv6 の両方に対応しているが、本実装では部分的に IPv4 のみ対応している部分がある。

6.1. プロトコルと通信メッセージ

本節では、前章で設計したプロトコルの詳細と、そのプロトコルで利用される通信形式について述べる。

6.1.1 プロトコル

プロトコルは、Simple Mail Transfer Protocol (SMTP)[12] や Post Office Protocol (POP)[13] などと同様に、ASCII 形式で人間が容易に識別できる方法を用いた。

コマンドは、それぞれのコマンド識別文字列と、コマンドによっては引数を伴い、CR LF で終端する。また一部のコマンドでは、ゲートウェイシステムの応答に従って、追加の情報を送信が必要なものもある。追加情報の有無や形式については、各コマンド中で説明する。

HELLO コマンド

HELLO コマンドの書式を図 6.1 に示す。

HELLO コマンドは引数を伴わない。

```
HELLO \r\n
```

図 6.1 HELLO コマンド

ゲートウェイシステムは、認証要求を受け付け可能な場合、追加情報として認証要求を出すように促す。エンドノードシステムは促しに従い、認証要求を送信する。認証要求は、後述する「通信メッセージ」の形式で定義する。

通信メッセージの送信後には '.' だけの行を送信し、メッセージの終端を伝える。

CONNECT コマンド

CONNECT コマンドの書式を図 6.2 に示す。

```
CONNECT <他エンドノードの IP アドレス> [ポート番号, プロトコル種別] \r\n
```

図 6.2 CONNECT コマンド

CONNECT コマンドは、接続要求を行う相手である他エンドノードの IP アドレスを引数に伴う。また、ポート番号とプロトコル種別を引数とすることもできる。ポート番号を引数とした場合、必ずプロトコル種別も引数としなければならない。両者を省略した場合、接続の要求の範囲を限定せず、すべてのポート番号、すべてのプロトコル種別について許可するよう要求することを示す。

IP アドレスの代わりとしてホスト名は利用できない。DNS 名前解決はエンドノードシステムで行う必要がある。

ゲートウェイシステムは、CONNECT コマンドを発行した自エンドノードの認証が済んでおり、かつ接続要求を受け入れ可能な場合、他エンドノードを保護するファイアウォールを探索し、接続要求を転送する。

REQUEST コマンド

REQUEST コマンドの書式を図 6.3 に示す。

```
REQUEST \r\n
```

図 6.3 REQUEST コマンド

要求先のゲートウェイシステムは HELLO コマンドと同様に、接続要求を受け付け可能な場合、接続要求を出すように促す。要求元のゲートウェイシステムは促しに従い、接続要求を送信する。この接続要求は、後述する「通信メッセージ」の形式で定義する。

通信メッセージの送信後には '.' だけの行を送信し、メッセージの終端を伝える。

CONNECT コマンドと異なり、接続要求を求める相手の IP アドレスは引数ではなく、通信メッセージに記す。通信メッセージは証明書を用いて電子署名するため改ざんの検知ができるが、引数では改ざんを検知できないためである。

INFO コマンド

INFO コマンドの書式を図 6.4 に示す。

```
INFO <取得したい情報種別> \r\n
```

図 6.4 INFO コマンド

情報種別として、以下の 4 種類を定義した。

INFO HELLO 自エンドノードが HELLO コマンドで送信した認証要求の情報。認証後でなければならない

INFO CONNECT 自エンドノードが CONNECT コマンドで接続要求を行った IP アドレスの一覧。認証後でなければならない

INFO SYSTEM ゲートウェイシステムの動作概況

INFO VERSION ゲートウェイシステムのプログラム・バージョン

QUIT コマンド

QUIT コマンドの書式を図 6.5 に示す。

```
QUIT \r\n
```

図 6.5 QUIT コマンド

QUIT コマンドは引数を伴わない。

ゲートウェイシステムは、HELLO コマンドによってエンドノードシステムから送信され、ゲートウェイシステムが保持しているエンドノードとエンドユーザの情報を破棄し、制御用の接続を切断する。

6.1.2 通信メッセージの形式

ゲートウェイシステムとエンドノードシステム間、またゲートウェイシステム相互間で認証や接続の要求を行う際に用いる ASCII 形式の文書を、本提案機構では「通信メッセージ」と呼ぶ。通信メッセージは、各システムの以下にあける通信のすべてに用いられる。

- エンドノードシステムからゲートウェイシステムへの認証要求メッセージ
- ゲートウェイシステム相互間での接続要求メッセージ
- ゲートウェイシステムからエンドノードシステムへの接続要求メッセージ

エンドノードシステムからゲートウェイシステムへの認証要求メッセージ

エンドノードシステムからゲートウェイシステムへの認証要求に用いられる通信メッセージの形式について述べる。

CR LF で終端する 1 行で 1 つの情報を伝達する。大括弧が情報の大分類、イコール記号の左端が情報の名前、右側が情報の内容である。情報の種類は次の通りである。

mtype メッセージ種別

このメッセージの種類を示す。エンドノードシステムからゲートウェイシステムへ認証を要求する際のメッセージのメッセージ種別は client である。

date 発行日付

クライアントが認証要求を求め、このメッセージを発行した日付をグリニッジ標準時間 (GMT) で示す。日付は、ゲートウェイシステムの処理に用いる 1970 年 1 月 1 日 0 時 0 分 0 秒からの秒数と、人間がメッセージを解釈可能なように YYYY/MM/DD hh:nn:ss の形式を併記する。

expire 有効期限

エンドノードシステムが希望する、認証の有効期限を示す。形式は発行日付と同一である。

ipaddr IP アドレス

認証を要求する、エンドノードシステムが導入されたエンドノードの IP アドレスを示す。

port 接続要求のためのポート番号

エンドノードシステムが、ゲートウェイシステムから、他のエンドノードシステムからの接続要求を受信するために、接続を受け付けるポートを示す。

permit 転送許可の相手

エンドノードシステムが、認証後に要求する接続要求の転送を許可する相手先を示す。

認証要求メッセージの例を図 6.6 に示す。

図 6.6 は、通信メッセージの内容が以下の通りであることを示している。

(a) エンドノードシステムからゲートウェイシステムへの認証メッセージであること

(b) グリニッジ標準時間の 2005/01/14 14:44:13 に要求が発行されたということ

(c) 自エンドノードが、グリニッジ標準時間の 2005/01/15 14:44:13 に失効することを希望していること

```
[general]
mtype=client                                ← (a)
date=2005/01/14 14:44:13 (1105713853)       ← (b)
expire=2005/01/15 14:44:13 (1105800253)     ← (c)
ipaddr=163.221.167.13                       ← (d)
port=41009                                  ← (e)

[permit]
target=*                                    ← (f)
```

図 6.6 認証要求メッセージ

(d) 自エンドノードの IP アドレスが 163.221.167.13 であること

(e) 自エンドノードが他エンドノードからの接続要求のため待機するポートは 41009 番であること

(f) 接続要求を転送する他ファイアウォールを限定しないこと

エンドノードシステムからの認証要求のメッセージは、前述のように、エンドユーザの秘密鍵によって電子署名され、S/MIME 形式でカプセル化されるため、交換されるメッセージは結果として図 6.7 のような形となる。

ゲートウェイシステム相互間での接続要求メッセージ

エンドノードシステムからゲートウェイシステムへの接続要求を、他のゲートウェイシステムへ転送する際に用いられる通信メッセージの形式について述べる。

CR LF で終端する 1 行で 1 つの情報を伝達する点など基本的な書式はエンドノードシステムからゲートウェイシステムへの認証要求メッセージと同一であるが、情報の種類が異なる。情報の種類は次の通りである。

mtype メッセージ種別

このメッセージの種類を示す。ゲートウェイシステム相互間による接続要求を転送するメッセージのメッセージ種別は igw である。

date 発行日付

接続元のゲートウェイシステムが、このメッセージを発行した日付をグリニッジ標準

```
MIME-Version: 1.0
Content-Type: multipart/signed;
  protocol="application/x-pkcs7-signature";
  micalg=sha1; boundary="----885EB0B0FFF2DCDA0749C48D2052BA8D"

This is an S/MIME signed message

------885EB0B0FFF2DCDA0749C48D2052BA8D
[general]
mtype=client
date=2005/01/14 14:44:13 (1105713853)
expire=2005/01/15 14:44:13 (1105800253)
ipaddr=163.221.167.13
port=41009

[permit]
target=*

------885EB0B0FFF2DCDA0749C48D2052BA8D
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIIKvQYJKoZIhvcNAQcCoIIKrjCCCqoCAQExDjAMBggqhkiG9w0CBQUAMAsGCSqG
SIb3DQEHAaCCCZcwggL9MIIB5aADAgECAgEQMA0GCSqGSIb3DQEBBQUAMDYxCzAJ
… (途中省略) …
yrE1HMu4nCYvUNXGef9P0z6p12+uH07Remqc3xS72c9PgbjF8fqegz/yuRqqcioN
0gLqko61FadbWlwl9A+CXfQ=

------885EB0B0FFF2DCDA0749C48D2052BA8D--
```

図 6.7 S/MIME カプセル化後の認証要求メッセージ

時間 (GMT) で示す。日付は、人間が容易に読解可能な YYYY/MM/DD hh:nn:ss の形式と、時間計算が容易な 1970 年 1 月 1 日 0 時 0 分 0 秒からの秒数の双方を併記する。

expire 有効期限

接続元のゲートウェイシステムが希望する、接続要求の有効期限を示す。形式は発行日付と同一である。

from 接続元のエンドノードの IP アドレス

接続を要求する元の、エンドノードシステムが導入されたエンドノードの IP アドレスを示す。

to 接続先のエンドノードの IP アドレス

接続を要求する先の、エンドノードシステムが導入されたエンドノードの IP アドレスを示す。

port 接続先のサービスのポート番号

接続を要求する先の、サービスのポート番号を示す。

usercert エンドユーザの電子証明書

接続を要求する元の、エンドユーザの電子証明書を示す。X.509 形式の電子証明書であって、これを BASE64 で符号化したものでなければならない。詳細は後述する。

接続要求メッセージの例を図 6.8 に示す。

図 6.8 は、通信メッセージの内容が以下の通りであることを示している。

(a) ゲートウェイシステム相互間の認証メッセージであること

(b) グリニッジ標準時間の 2004/11/01 12:34:56 に要求が発行されたということ

(c) 自エンドノードが、グリニッジ標準時間の 2004/11/02 12:34:56 に失効することを希望していること

(d) 接続を行う自エンドノードの IP アドレスが 192.168.2.2 であること

```
[general]
mtype=igw                                   ← (a)
date=2004/11/01 12:34:56 (1102483660)       ← (b)
expire=2004/11/02 12:34:56 (1102570060)     ← (c)
from=192.168.2.2                            ← (d)
to=192.168.1.2                              ← (e)
port=80                                     ← (f)

[user-cert]                                 ← (g)
MIIDJjCCAg6gAwIBAgICC6swDQYJKoZIhvcNAQEFBQAwPjELMAkGA1UEBhMCSlAx
FTATBgNVBAoTDFdJREUgUHJvamVjdDEYMBYGA1UECxMPbWVtYmVycyBvbmx5IENB
… (途中省略) …
SNhrpdRcTCbZCOKvIGH+ixdfUpFE807JKtnQ+MgruVfKNpduIh8RF2H1tElr7aoh
AMcblvquXE2H/1oKZ7OU+Dvg+UjDT8uDOsjpXTOvS1O5Vi2jL4Ndcvmy
```

図 6.8 ファイアウォール間で転送される接続要求メッセージ

(e) 接続を行いたい他エンドノードの IP アドレスが 192.168.1.2 であること

(f) 接続を行いたいポートが 41009 番であること

(g) エンドユーザの証明書

ゲートウェイシステム相互間の接続要求のメッセージは、ゲートウェイシステムの管理者の秘密鍵によって電子署名されるため、交換されるメッセージは結果として図6.9 の形になる。

```
MIME-Version: 1.0
Content-Type: multipart/signed;
  protocol="application/x-pkcs7-signature";
  micalg=sha1; boundary="----45D27DF450A9EECC5ACE59F6690DAAEF"

This is an S/MIME signed message

------45D27DF450A9EECC5ACE59F6690DAAEF
[general]
mtype=igw
date=2004/11/01 12:34:56 (1102483660)
expire=2004/11/02 12:34:56 (1102570060)
from=192.168.2.2
to=192.168.1.2
port=80

[user-cert]
MIIDJjCCAg6gAwIBAgICC6swDQYJKoZIhvcNAQEFBQAwPjELMAkGA1UEBhMCSlAx
FTATBgNVBAoTDFdJREUgUHJvamVjdDEYMBYGA1UECxMPbWVtYmVycyBvbmx5IENB
… (途中省略) …
SNhrpdRcTCbZCOKvIGH+ixdfUpFE807JKtnQ+MgruVfKNpduIh8RF2H1tElr7aoh
AMcblvquXE2H/1oKZ7OU+Dvg+UjDT8uDOsjpXTOvS1O5Vi2jL4Ndcvmy

------45D27DF450A9EECC5ACE59F6690DAAEF
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIIFAAYJKoZIhvcNAQcCoIIE8TCCBO0CAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3
DQEHAaCCAyowggMmMIICDqADAgECAgILqzANBgkqhkiG9w0BAQUFADA+MQswCQYD
… (途中省略) …
1FFjHwAcYswlCv7AD+TY88B1kSQh7+UFbZpI7PcUiV2JBjf7+VlHjc6DLkwWwkrN
3RShNYOAOm1/Z6nXacc2adTonj00aPKyl9R7e8oEt5bhx3No

------45D27DF450A9EECC5ACE59F6690DAAEF--
```

図 6.9 S/MIME カプセル化後の接続要求メッセージ

6.2. ゲートウェイシステム

ゲートウェイシステムは、表 6.1 に示す環境で実装した。

表 6.1 ゲートウェイシステムの実装環境

OS	OpenBSD 3.6
言語	C 言語
コンパイラ	gcc 2.95.3
ライブラリ	OpenSSL (libssl) 0.9.7d

このゲートウェイシステムの実装が想定する動作環境は、OpenBSD 3.4 以降である。

ゲートウェイシステムの実装では、実際にパケットフィルタリングを行うファイアウォールの機構として OpenBSD 3.6 の pf を用いた。pf は柔軟性が高く、また他の環境への移植も積極的に進められていることから、今後広く普及すると考えたためである。メッセージの署名や検証といった証明書の処理には OpenSSL 0.9.7d の libssl および libcrypto を用いた。

本節では、ゲートウェイシステムの実装について、機能ごとに説明する。

6.2.1 署名済みメッセージの検証

HELLO コマンドではエンドノードシステムの認証のため、REQUEST コマンドではゲートウェイシステムの認証のため、メッセージの電子署名の検証を行う。

本機構では、メッセージは S/MIME のクリアテキスト署名の形式を用いるため、メッセージは本体と書名部分に分離できる。検証のためには、まずこの両者を分離しなければならない。libssl には S/MIME の取り扱いのための補助関数が用意されており、分離にはこの補助関数を用いる。また検証にも libssl の PKCS7_verify 関数を用いる。

HELLO コマンドでは、メッセージの検証が成功した場合、電子署名からエンドユーザの証明書を取り出し、メッセージの内容であるエンドユーザの情報と関連づけて、エンドユーザ情報リストとして記憶する。REQUEST コマンドでは、特に証明書の記憶は行わない。

6.2.2 相手先ノードを保護するファイアウォールの検出

CONNECT コマンドを受け付けた場合、その接続要求を、接続を求める相手先ノードを保護下におくファイアウォールに転送する。このため、そのファイアウォールを検出する必要がある。

今回はプロトタイプの実装であるため、容易に実装が可能なものとして、今回は ICMP ECHO メッセージを用いた。traceroute と同様に、TTL を 1 から 1 ずつ増加させたパケットを送信し、接続を求める相手先ノードの、一つ前にあるノードを保護するファイアウォールとみなした。

6.2.3 接続要求の転送

CONNECT コマンドによる接続要求を、検出したファイアウォールに転送する。

接続要求を行った自エンドノードについては、HELLO コマンドで認証を行った際に、エンドユーザの証明書を記憶している。自エンドノードからの接続要求を他ファイアウォールに転送する際には、現在保持しているエンドユーザ情報リストを探索し、当該のエンドノードの利用者の証明書を添付し、転送する。

転送のために、TCP を使ってファイアウォールに接続し、REQUEST コマンドを用いる。

6.2.4 ファイアウォール制御

ゲートウェイシステムは、以下の状態で、ファイアウォールを制御する。

- 自エンドノードからの CONNECT コマンドによる接続要求を転送した結果、他エンドノードから他ファイアウォールを経由して許可という返答を得た場合、自エンドノードからの要求に応えられるよう、ファイアウォールに pass out のルール設定を行う。

- 他ファイアウォールから転送された、他エンドノードからの接続要求を自エンドノードに転送した結果、自エンドノードから許可という返答を得た場合、他エンドノードからの要求に応えられるよう、ファイアウォールに pass in のルール設定を行う。

ファイアウォールのルール設定は、設定ファイルである pf.conf を変更することで行う。具体的に設定するルールは以下の通りである。複数行にまたがっているもので、行末が矢印（→）のものは、実際は一行で記す。

pass out 設定 (TCP)

pass out proto tcp from（**自エンドノードの** IP **アドレス**）→
to（**他エンドノードの** IP **アドレス**）port（**希望するポート番号**）→
flags S/SA keep state

pass out 設定 (UDP)

pass out proto udp from（**自エンドノードの** IP **アドレス**）→
to（**他エンドノードの** IP **アドレス**）port（**希望するポート番号**）

pass in 設定 (TCP)

pass in proto tcp from（**他エンドノードの** IP **アドレス**）→
to（**自エンドノードの** IP **アドレス**）port（**希望するポート番号**）→
flags S/SA keep state

pass in 設定 (UDP)

pass in proto udp from（**他エンドノードの** IP **アドレス**）→
to（**自エンドノードの** IP **アドレス**）port（**希望するポート番号**）

設定に先立ち、すでに同じルール設定がされていないかどうかを検出し、設定されている場合は変更を行わない。変更を行った後は、pfctl を用いて pf に設定ファイルの変更を通知する。

TCP の設定の際の ”flags S/SA keep state” は、TCP においてファイアウォールのステートフル・インスペクションを用いるためのキーワードである。”flags S/SA” とは、SYN フラグまたは SYN/ACK フラグの立った TCP パケットのみ適合させることを示す。”keep state” によって、それらパケットに基づいて TCP の状態を保持し、その TCP コネクションのパケットをすべて通過させることを示す。

ゲートウェイシステムは pthread を用いてマルチスレッドで動作しており、ソケット通信の処理は複数の要求を平行して処理することができる。しかし、設定ファイルの操作は並列して行うことはできない。このため、実際に設定を行う関数は pthread の mutex を用いて実行を直列化 (serialize) する。

mutex によって直列化される処理はファイル操作だけである。ファイル操作はソケット通信の処理と異なり、処理にかかる時間はほとんど一定であり、また実際の I/O の処理は大規模なものにはならないため、この直列化が全体の処理に与える影響は軽微であると考える。

6.3. エンドノードシステム

エンドノードシステムは、表 6.2 に示す環境で実装した。

表 6.2 エンドノードシステムの実装環境

OS	Windows XP
言語	C 言語
開発環境	Microsoft Visual Studio .NET 2003
コンパイラ	Microsoft 32-bit C/C++ Compiler 13.10.3077
ライブラリ	Microsoft CryptoAPI (CAPI)

このエンドノードシステムの実装が想定する動作環境は、Windows XP である。

ゲートウェイシステムと異なり、エンドノードシステムの実装は、エンドノードで最も用いられている Windows 環境で動作するものとした。また証明書の処理も Windows 標準の Microsoft CryptoAPI を用いた。これは、ゲートウェイシステムで用いた OpenSSL 以外のライブラリを用いて実装することで、特定のライブラリに依存せずとも実装が可能であることを示すためである。

本節では、エンドノードシステムの実装について、機能ごとに説明する。

6.3.1 メッセージ署名

エンドノードシステムは、自エンドノードとエンドユーザの情報から、認証要求のメッセージを生成する。

エンドノードは、認証要求メッセージをエンドユーザの秘密鍵を用いてクリアテキスト署名を行う。S/MIME 形式では通常の電子署名の場合は本文のエンコード形式が BASE64 になるが、クリアテキスト署名の場合は本文のエンコードは変更されない。また、電子署名にはエンドユーザの証明書を含む。

署名には、Windows の CryptoAPI の CryptoSignMessage API を用いる。

6.3.2 アプリケーション・インタフェース

エンドノードのアプリケーションは、アプリケーションに組み込むライブラリを用いて、エンドノードシステムのプロセスに対して、プロセス間通信によって接続要求を伝達する。具体的には、プロセス間通信には Windows メッセージングシステムを用いる。

Windows メッセージングシステムでは複数のアプリケーションからの要求を同時に受け付けることが可能である。エンドノードシステムは、メッセージ受信用のウィンドウを作成し、アプリケーションはそのウィンドウに対して Windows メッセージを送信する。要求は自動的に直列化され、エンドノードは要求を逐次的にゲートウェイシステムに送信する。

アプリケーション・インタフェースの関数を表 6.3 にあげる。

表 6.3 アプリケーション・インタフェースの関数

FeitInitialize	アプリケーション・インタフェース初期化関数
FeitConnect	接続を要求する関数
FeitDisconnect	接続の解除を要求する関数

FeitInitialize は、アプリケーション・インタフェース初期化関数である。エンドノードシステムの Windows メッセージ受信用ウィンドウの検索を行う。アプリケーション・インタフェースの使用前に必ず呼び出す必要がある。

FeitConnect は、アプリケーションが他エンドノードへ接続する際に呼び出す接続要求関数である。アプリケーション・インタフェース・ライブラリは、FeitInitialize で発見したエンドノードシステムの WIndows メッセージ受信用ウィンドウへメッセージを送信し、応答を待つ。

FeitDisconnect は、アプリケーションが FeitConnect によって接続を行った他エンドノードと切断する際に呼び出す接続の解除要求関数である。FeitConnect と同様に、アプリケーション・インタフェース・ライブラリは、FeitInitialize で発見したエンドノードシステムの Windows メッセージ受信用ウィンドウへメッセージを送信し、応答を待つ。

アプリケーション・インタフェース・ライブラリは永続して使用するメモリの確保を行わないため、初期化関数に対応する終了処理関数は存在しない。

6.3.3 接続要求の可否判断

エンドノードシステムは、他エンドノードからの接続要求メッセージの許否を判断する。

接続要求には他エンドノードの利用者の証明書が添付されており、また他エンドノードを保護下に置くファイアウォールの管理者によって署名がなされている。これらの情報を元に、エンドノードシステムはその証明書の情報を元に判断を行う。

本実装での判断条件は、エンドユーザが指定した証明書の項目で、指定した文字列と一致するか否かを判断する。たとえば、証明書の発行元に “Some Project” という文字列が含まれていれば許可とし、含まれていない場合は拒否とする。

第7章　実験と評価

本章では、本提案の有用性を明らかにするため、前章の実装を用いた実験を行い、その結果をもとに提案と提案機構の評価を行った。

7.1. 実験環境

4台のPCを用いて実験ネットワークを構築した。そのうち2台がファイアウォールであり、もう2台がエンドノードである。ファイアウォールをそれぞれファイアウォールA、ファイアウォールBとし、ファイアウォールAが保護するノードをエンドノードA、ファイアウォールBが保護するノードをエンドノードBとする。

ファイアウォールA、Bの性能を表7.1に、エンドノードA、Bの性能を表7.2に示す。

表 7.1　ファイアウォールPCの性能

	ファイアウォールA	ファイアウォールB
CPU	Pentium II 300MHz	Pentium II 401MHz
RAM	256MB	128MB
NIC (ゲートウェイ相互間)	3Com 3c905 100Base-TX	Intel 82557
IPアドレス (〃)	163.221.167.11	163.221.167.12
NIC (エンドノード向け)	Intel 82557	Intel 82557
IPアドレス (〃)	192.168.1.1	192.168.2.1
HDD	IBM-DHEA-38451	IBM DTTA-351010
OS	OpenBSD 3.6	OpenBSD 3.6

各ファイアウォールとエンドノード間はクロスケーブルで接続した。ファイアウォール相互間の接続にはNETGEAR GSM712ギガビット・スイッチングハブを用いた（ただし、PC側のネットワークインタフェースは100Base-TXである）。経路の設定は、route

表 7.2 エンドノード PC の性能

	エンドノード A	エンドノード B
CPU	Pentium III 500MHz	Pentium 4 2.53GHz
RAM	128MB	1024MB
NIC (ゲートウェイ向け)	Intel 82558	Realtek RTL8139
IP アドレス (〃)	192.168.1.2	192.168.2.2
HDD	Seagate ST36421A	Seagate ST380011A
OS	Windows XP SP2	Windows XP SP2

コマンドによって静的に行った。この実験ネットワークのトポロジ図を図 7.1 に示す。

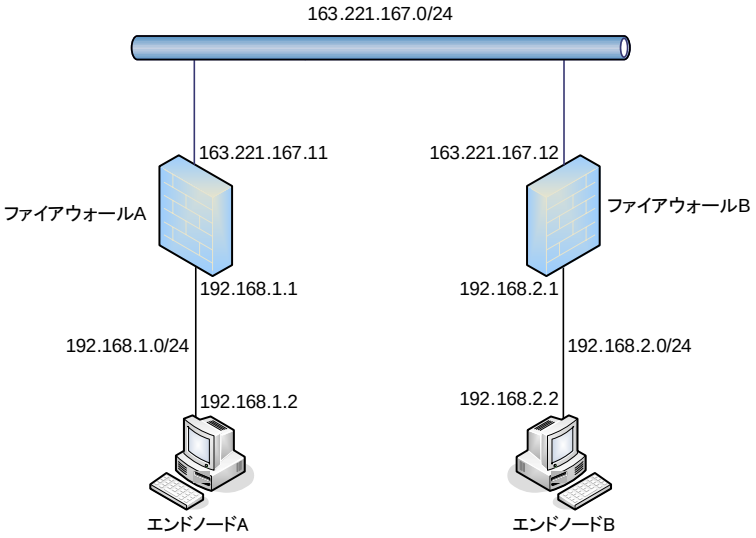

図 7.1 実験ネットワークのトポロジ図

実験では、エンドノード A を接続を受け付ける側、エンドノード B を接続を行う側とした。エンドノード A、エンドノード B のそれぞれに提案機構のエンドノードシステムを導入し、またエンドノード A には計測のためのサーバアプリケーションとして Apache 1.3.33 を導入した。

7.2. 実験結果

実験では、提案機構を用いることによるスループットへの影響と、処理に要する時間について計測した。また、実験で得られた結果から、本提案が P2P ネットワークの実現を容易にできたかどうかについて考察する。

7.2.1 スループットに与える影響

スループットの測定には、HTTP を用いた。HTTP を用いて Web サーバから異なるサイズのファイルを取得するのに要した時間をそれぞれ測定し、スループットを求めた。また、スループットへの影響が、ファイアウォールを用いたことによるものか、提案機構によるものかを明確にするため、表 7.3 の 4 種類の状態で計測した。

表 7.3 評価を行った状態

without-pf	ファイアウォール (pf) を無効にした状態
with-pf	ファイアウォール (pf) を有効にして、設定が空の状態
with-pf-rule	ファイアウォール (pf) を有効にして、 実験に用いる通信を許可するルール設定を行った状態
with-feit	提案機構を有効にした状態

ファイルサイズは、1, 2, 5, 10, 20, 50, 100 メガバイトの 7 種類とした。

それぞれのファイルに対して 100 回試行を繰り返し、その平均を用いた。

表 7.4 に、転送に要した時間を示した。表 7.5 では、スループットを示した。表 7.6 では、提案機構による処理のオーバーヘッドを示すため、ファイアウォールのルール設定後の状態 (with-pf-rule) を 1 とした、相対的なスループットを示した。

表 7.5 をグラフにしたのが図 7.2 である。また、表 7.6 をグラフにしたのが図 7.3 である。

表 7.4 実験結果: 転送に要した時間

状態	転送に要した時間 [ミリ秒]						
	1MB	2MB	5MB	10MB	20MB	50MB	100MB
without-pf	259.36	490.94	1105.94	2109.38	4202.34	10452.81	20905.15
with-pf	296.87	536.41	1220.32	2372.65	4648.44	11590.46	23189.06
with-pf-rule	320.15	598.28	1264.69	2445.78	4782.98	11841.40	23740.46
with-feit	739.21	967.34	1628.14	2815.94	5198.60	12218.90	24032.66

表 7.5 実験結果: スループット

状態	スループット [M ビット/秒]						
	1MB	2MB	5MB	10MB	20MB	50MB	100MB
without-pf	30.845	32.591	36.168	37.926	38.074	38.267	38.268
with-pf	26.948	29.828	32.778	33.718	34.420	34.511	34.499
with-pf-rule	24.988	26.743	31.628	32.709	33.452	33.780	33.698
with-feit	10.822	16.540	24.568	28.410	30.778	32.736	33.288

表 7.6 実験結果: with-pf-rule と比較した際の相対的なスループット

状態	with-pf-rule を 1 とした場合の比						
	1MB	2MB	5MB	10MB	20MB	50MB	100MB
without-pf	1.2344	1.2186	1.1435	1.1595	1.1382	1.1328	1.1356
with-pf	1.0784	1.1153	1.0364	1.0308	1.0289	1.0217	1.0238
with-pf-rule	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
with-feit	0.4331	0.6185	0.7768	0.8685	0.9201	0.9691	0.9878

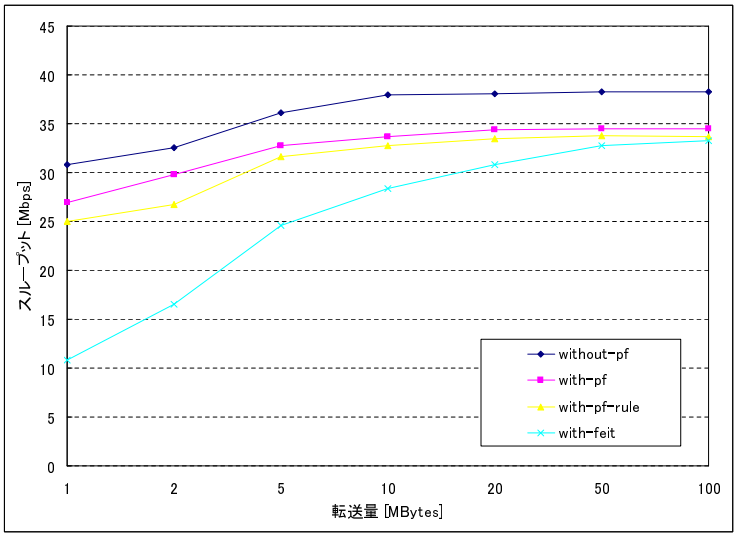

図 7.2 実験結果: スループット

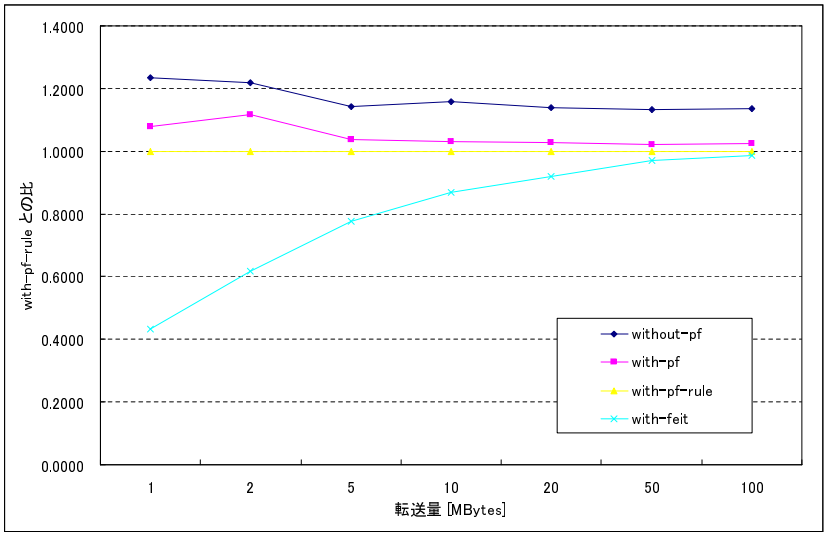

図 7.3 実験結果: with-pf-rule と比較した際の相対的なスループット

7.2.2 処理のオーバーヘッド

提案機構のゲートウェイシステムにおいて、各処理部ごとの処理時間を計測した。認証要求の処理、接続要求の処理、切断など一連の処理を 5 回試行し、その平均を求めた。

接続を受け付ける、エンドノード A 側のファイアウォール A における各部の処理時間を表 7.7 に示した。接続を行う、エンドノード B 側のファイアウォール B における各部の処理時間を表 7.8 に示した。処理時間は malloc などを含めて呼び出す関数ごとに求めたが、表では関係する関数を機能のブロックとしてまとめた。また、処理時間が 0.5msec 以下のものは影響が小さいと判断し、省略した。

表 7.7 実験結果: ファイアウォール A における各部の処理時間

ソケットからの読み込み	161.024 msec.
ファイアウォールの制御	51.167 msec.
通信メッセージの署名の検証	30.915 msec.
TCP 接続	0.714 msec.
ソケットへの書き込み	0.643 msec.
通信メッセージの解釈	0.617 msec.

表 7.8 実験結果: ファイアウォール B における各部の処理時間

接続要求の転送	329.480 msec.
通信メッセージの署名の検証	29.400 msec.
ファイアウォールの制御	28.140 msec.
相手先ノードを保護するファイアウォールの検出	2.360 msec.

7.3. 結果に関する考察

実験によって得られた結果について考察する。

提案機構を用いた場合、転送量が小さい場合に、スループットに与える影響が顕著である。1 メガバイトの転送の場合、スループットは半分以下となる。提案機構ではデータ

転送に先立ち、ファイアウォールを制御するための通信を行うため接続にかかる時間が増加する。

今回の実験においては、接続に要する処理のオーバーヘッドは 300 ミリ秒から 400 ミリ秒であった。1 メガバイトの転送に要する時間は、ファイアウォールが無効の状態で 220 ミリ秒、ファイアウォールのルール設定後の状態でも 320 ミリ秒であるため、400 ミリ秒の処理は非常に大きな影響がある。しかし 100 メガバイトの場合、転送に要する時間が 22,000 ミリ秒から 23,700 ミリ秒となり、提案機構の影響は 1.2%程度であり、誤差の範囲となる。

また、提案機構のゲートウェイシステムにおいて各処理部ごとの処理時間を計測した結果、以下の部位で処理に時間を要することが分かった。

- 認証要求、接続要求の通信処理（主に、ソケットからの読み込み）
- 通信メッセージの署名の検証
- 相手先ノードを保護するファイアウォールの検出
- ファイアウォールの制御

接続要求に関しては、自ファイアウォールから他ファイアウォールへの接続と、他ファイアウォールから他エンドノードへの接続の 2 つが行われるため、その通信処理の遅延が顕著である。ソケットからの読み込みを行単位で処理するための関数呼び出しに 160 ミリ秒以上かかっているが、これは実装上の問題であり、読み込みの単位を変更することで改善できる可能性がある。

署名の検証やファイアウォールの制御といった作業は、それぞれ 30 ミリ秒から 50 ミリ秒程度である。署名の検証は並列処理が可能であるが、ファイアウォールの制御は直列化が必要であるため、アクセスの集中で遅延が生じる。しかし、ファイアウォール制御の方法を設定ファイル経由でなく ioctl を用いてブロックデバイスの /dev/pf を直接制御することで、少なくともディスク I/O が不要となり、また直列化が不要となる可能性がある。

P2P アプリケーションでは、情報の検索などの目的では小規模の転送が行われるが、実際の情報の転送は大容量であることが多い。また一度接続を行うと接続を維持するこ

とが少なくない。提案機構では接続後のスループットへの影響は通常のファイアウォールのルール設定と同程度であるため、接続を長時間維持する P2P モデルの設計を行うことで、影響を軽微にできる。

第8章　結論

本論文では、P2P モデルを実現するための相互接続性を確保する方法として、エンドノードの主導によってファイアウォールを動的に再構成するファイアウォール制御技術について提案した。また、ファイアウォールを制御し、通信を可能とする上で必要なネットワークやノードの安全性を維持するための方法として、エンドユーザに立脚した、エンドユーザ同士による信頼モデルについて提案した。

また、その提案に基づいた機構を実際に設計、実装し、有効性を検証するための実験を行った。結果として、以下のことが明らかとなった。

- 提案機構を用いることで、エンドユーザの主導によるファイアウォール制御を実現できた
- 提案機構による処理遅延は、転送量とは独立である
  したがって、転送量が小さい場合は提案機構によるオーバーヘッドが大きいが、転送量が大きい場合は無視できる
- 接続を長時間維持する P2P モデルでは、提案機構はより有効に機能する
- 処理遅延は、実装上生じている部分がある
  大部分の処理遅延はソケット通信において生じており、効率のよい実装によって改善する可能性がある

また、今後の課題として以下のことがあげられる。

相手先ノードを保護するファイアウォールの検出手法

相手先ノードを保護するファイアウォールの検出に、本実装では ICMP ECHO メッセージを用いた。これは、ファイアウォールの他の機能に影響を与えず、OS のカーネル部分に修正を加えずに済むという移植性を考慮したものだが、この方法では検出に時間がかかり、また本機構に対応していないルータに対しても負荷がかかる。

さらに、多段のファイアウォールへの対処が行えないなど、問題が多い。本提案を実現するには、移植性が高く、ファイアウォールを高速に検出が可能で、かつ提案機構に対応しないファイアウォールへの影響を低く抑えることが可能な手法が必要である。

アプリケーション・インタフェース

本実装はプロトタイプ実装であるため、アプリケーション・インタフェースの利便性についてはほとんど考慮していない。しかし P2P アプリケーションの容易な実現のためには、アプリケーション側の対応が不要となるような仕組みを考慮すべきである。Windows の場合は API のフック技術、UNIX 系 OS の場合はライブラリのラッパー技術などが候補となる。

接続要求の許否判断のポリシー

接続要求に際して、要求を行う先のエンドノードへ、証明書を用いたエンドユーザの情報の提供を行う機構は提供できたが、接続要求を許否するポリシーについてはより考慮が必要である。本実装での判断条件は、エンドユーザが指定した項目で、指定した文字列と一致するか否かという単純なものとしたが、どの程度柔軟な指定が可能であれば過不足なく許否の判断が行えるかという点には議論の余地がある。

検疫システムとの連携

本提案は、エンドノードとエンドユーザの情報に基づいて、接続要求の許否を判断するが、現状では利用が容易な検疫システムがなかったため、エンドノードの情報については利用していない。今後、検疫システムから情報を取得するための手段は標準化されると考えるが、その場合は、接続要求の許否の判断にエンドノードの情報を用いることが可能となるよう、検疫システムの情報を活用する機構が必要であろう。

提案機構は様々な課題を残しているが、P2P モデルの実現のために相互接続性を確保する方法として、エンドノード主導によるファイアウォール制御は一定の効果をもたらすことを示し、また将来における可能性を示すことができた。

謝辞

本研究を進めるにあたり、多くの方々に暖かいご指導、貴重な助言をいただきました。末尾ではありますが、この場を借りて皆様にお礼申し上げます。

まずはじめに、指導教官として研究を指導していただいた奈良先端科学技術大学院大学情報科学研究科インターネット・アーキテクチャ講座の砂原秀樹教授に心よりの感謝を申し上げます。

副指導教官として貴重な助言をいただいた奈良先端科学技術大学院大学情報科学研究科インターネット工学講座の山口英教授に心から感謝申し上げます。

研究をすすめるにあたって貴重なご教示をいただいた奈良先端科学技術大学院大学情報科学研究科インターネット・アーキテクチャ講座の藤川和利助教授に心から感謝申し上げます。

充実した研究環境を提供していただき、研究の上で、また論文執筆の上で多くのご指摘をいただいた奈良先端科学技術大学院大学情報科学センターの中村豊助手、垣内正年助手に感謝申し上げます。また、研究をすすめるにあたって様々なご教示をいただいたインターネット工学講座の和泉順子女史、ならびにインターネット・アーキテクチャ講座の市川本浩氏に感謝申し上げます。

研究、また研究室生活にあたって様々な事務処理をしていただいた事務補佐員の能勢佳苗女史に感謝申し上げます。

研究にあたって様々な相談に乗っていただき、また研究生活を支えてくださったインターネット・アーキテクチャ講座の学生のみなさまに感謝いたします。

最後に、研究生活をさまざまな側面から支えてくれた家族と友人たちに感謝します。この論文の半分は、彼らの支えでできています。

参考文献

[1] Internet System Consortium, “Internet Domain Survey, Jul 2004.”, http://www.isc.org/index.pl?/ops/ds/, July 2004.

[2] K. Egevang, P. Francis, “The IP Network Address Translator (NAT)”, Request For Comments 1631, Internet Engineering Task Force (IETF), May 1994.

[3] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, Request For Comments 2460, Internet Engineering Task Force (IETF), December 1998.

[4] 白井 出, “DNS Day:JP DNS Updates”, http://jprs.jp/tech/material/IW2004-DNS-DAY-JPDNS-shirai.pdf, Internet Week 2004 DNS DAY 資料, December 2004.

[5] 池田 信夫, 山田 肇, “IPv6 は必要か”, http://www.rieti.go.jp/jp/publications/summary/02012500.html, 経済産業研究所ディスカッションペーパー, January 2002.

[6] W. Diffie, M.E. Hellman, ”New directions in cryptography”, IEEE Trans. On Information Theory, Vol.IT-22, No.6, pp.644-654, November 1976.

[7] “Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks”, International Telecommunication Union, March 2000.

[8] R. Housley, W. Ford, W. Polk, D. Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, Request For Comments 2459, Internet Engineering Task Force (IETF), January 1999.

[9] “OpenSSL Project”, http://www.openssl.org/

[10] "RSA Security, Inc." http://www.rsasecurity.com/rsalabs/

[11] "Public Key Cryptography Standards", http://www.rsasecurity.com/rsalabs/pkcs/, RSA Security, Inc.

[12] J. Klensin, "ISimple Mail Transfer Protocol", Request For Comments 2821, Internet Engineering Task Force (IETF), April 2001.

[13] J. Myers, M. Rose, "Post Office Protocol - Version 3", Request For Comments 1939, Internet Engineering Task Force (IETF), May 1996.