社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－122（38）
2005－CSEC－28（38）
2005／3／23

エンドノード主導によるファイアウォール制御機構

鬼丸　敬輔 †　和泉　順子 †　市川　本浩 †　砂原　秀樹 ††

近年，P2P(Peer-to-Peer) 型通信モデルに注目が集まっている．しかし，P2P 型通信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．相互接続性を確保するための技術として仮想ネットワーク技術などがあるが，ネットワーク管理者が利用者を事前に，手動で設定する必要があり，P2P 型通信モデルに好適とは言えない．本稿では，相互接続性を確保し，P2P 型通信モデルを実現するために，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機構について提案する．またファイアウォール制御にあたって必要となる，通信の許否を判断するためのエンドノード相互間における信頼モデルについて提案する．

Design of End-node Oriented Firewall Traversal

Keisuke Onimaru†　Michiko Izumi†　Motohiro Ichikawa†　Hideki Sunahara††

Peer-to-peer communication model was attracted interest. Unfortunately, the model requires all end-node to be interconnectable. Technologies to keep interconnectivity and to put peer-to-peer communication model into practice include technologies for firewall traversal, virtual private network, and so on. Before using these technologies, administrator needs to permit access. As a consequence, there is insufficiency for p2p communication model. In this paper, we proposed the end-node oriented firewall control mechanism for keep interconnectivity, and to realize the peer-to-peer communication model. And we proposed the end user oriented mutual trust model for applying this technology.

1　はじめに

現在のインターネットではクライアント・サーバ型通信モデルが一般に用いられている．しかし利用者や，取り扱われる情報の増加，家庭向け光ファイバ回線などのアクセス回線の高速化によって，サーバ側の負荷の増大が問題となっている．また，クライアント・サーバ型通信モデルではプル型の通信を用いる必要がある．しかしプル型の通信ではサービスを受けるために能動的に接続しなければならないことが多く，接続しない限り情報の更新を知ることができないため，情報の伝達が遅れるという問題がある．

これらの問題の解決のために，P2P(Peer-to-Peer) 型通信モデルが注目されている．P2P 型通信モデルではすべてのノードが，クライアント・サーバ型通信モデルにおけるサーバとクライアントの役割を兼ねるため，特定のノードへ負荷が集中することを抑制できる．またプッシュ型の通信を用いることが可能であるため，情報伝達の遅延を低減させることができる．しかし P2P 型通信モデルの実現には，ノード同士が相互に透過的に接続可能である必要がある．

インターネットに接続する組織や企業では，悪意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の利用が一般的であり，相互接続性は保証できない．また今後も悪意ある通信が減少する見込みは少なく，ファイアウォールは現状，欠かすことができない [2]．このことから，相互接続性を確保し，P2P 型通信モデルを実現するために，ファイアウォールと共存可能な技術が望まれている．

相互接続性を確保するための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いる．しかしこれらの技術には様々な問題がある．ファイアウォール透過技術では，代理サーバを設置し接続を仲介するが，代理サーバの負荷が大きくスループットの低下が著しい．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．またこれら

† 奈良先端科学技術大学院大学 情報科学研究科
Nara Institute of Science and Technology, Graduate School of Information Science
†† 奈良先端科学技術大学院大学 情報科学センター
Nara Institute of Science and Technology, Information Science Center

の技術に共通する問題として，利用者の追加や削除といった設定は，管理者が事前に，手動で行わなければならないという点がある．このため利用者が増減するたびにネットワーク管理者は設定を行わねばならず，多数のノードを接続することは現実的に困難である．そのためこれらの技術は，相互接続性を確保することはできるが，P2P 型通信モデルの実現に好適であるとは言えない．

そこで，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機構について提案する．本提案は，エンドノードの要求にしたがって，ファイアウォールがエンドノードの設定やエンドユーザの能力に応じて，ファイアウォールを再構成する．加えて，エンドユーザ間で相互に接続が可能なように，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を相互に連携可能とする．

本稿では，2 節で相互接続を確保するための既存技術について説明し，3 節で提案について述べ，4 節で提案機構の設計を示す．5 節で設計に基づいたプロトタイプの実装を示し，またその実装を用いた実験と評価について検討する．最後に 6 節で本稿のまとめを行う．

2　相互接続性を確保するための既存技術

本節では，相互接続性を確保するための既存技術として，ファイアウォール透過技術の SOCKS と仮想ネットワーク技術の L2TP/IPsec VPN について概要を述べる．

2.1　SOCKS

SOCKS[1] は，OSI 参照モデルの第 7 層による，ファイアウォールを透過するための技術である．ファイアウォールの内部と外部の境界にアプリケーションゲートウェイを設置し，このゲートウェイが IP パケットを中継する．ファイアウォールの内部と外部をまたいで通信を行いたい場合，まず代理サーバに接続し，代理サーバが接続を中継することで機能する．この方法によってファイアウォールの内部と外部の接続が可能となるが，一度受信した通信を再送信するため，通信スループットの低下が著しい．

2.2　L2TP/IPsec VPN

L2TP(Layer 2 Tunneling Protocol)[4]/IPsec VPN[5] は，OSI 参照モデルの第 2 層および第 3 層によって仮想ネットワークを構築するための技術である．ファイアウォールの内部に VPN ゲートウェイを設置し，このゲートウェイが IPsec を用いてプライベートネットワークのパケットをカプセル化し，物理的なネットワークの IP パケットとして他の VPN ゲートウェイと交換することで機能する．この方法によってファイアウォールで遮蔽された，異なるネットワークにあるノード同士が相互に接続可能となる．しかし L2TP/IPsec VPN は信頼されたノード同士による閉じたネットワーク構築の技術であるため，プライベートネットワーク間にファイアウォールは設置されず，適切な保護を受けるのが困難である．また，暗号化のため物理ネットワークの管理者が通信を把握できないという問題もある．

2.3　既存技術のまとめ

本節で取り上げた既存に共通する問題として，利用者の追加や削除といった設定は，管理者が事前に，手動で行わなければならないという点がある．このため利用者が増減するたびにネットワーク管理者は設定を行わねばならず，多数のノードを接続することは現実的に困難である．

したがってこれらの既存技術では，相互接続性を確保することは可能であるが，P2P 型通信モデルに好適であるとは言えない．

3　エンドノード主導のファイアウォール制御機構の提案

ファイアウォールと共存可能な，相互接続性を確保するための機構について提案する．提案はエンドノード主導によるファイアウォール制御機構と，ファイアウォール制御のためのエンドノード相互間の信頼モデルの 2 つの要素から成る．

3.1　用語の定義

本節以降で用いる用語を次のように定義する．

エンドユーザ: 一般の利用者のこと

エンドノード: エンドユーザが利用する，パケットのルーティングを行わない末端のノードのこと

ゲートウェイ: パケットのルーティングを行う，ファイアウォールの導入されたノードのこと

ファイアウォール: ゲートウェイに導入される通信制御のための機構のこと

3.2 エンドノード主導のファイアウォール制御機構

現在のファイアウォールはネットワーク管理者によって制御されており，保護するネットワークの最も弱点となるノードにあわせて一律に，強力な保護を行うのが一般的である．このため，悪意ある通信に十分対処可能なノードでは保護が過剰となり，結果として相互接続性を確保できていない．

そこで，エンドノード，ひいてはエンドユーザの主導のもとにファイアウォールを制御する機構について提案する．提案機構によって，悪意ある通信へ十分対処可能で安全と認められるノードに関しては，個別に保護レベルを設定することで制御を緩和し，それによって相互接続性を確保することが可能となる．

エンドノードは希望する通信に応じて，ファイアウォールにあらかじめ設定されたポリシーの元でファイアウォールを制御する．ファイアウォールの管理者は，たとえばワームの流行によって特定のノードやポートへの攻撃が行われている場合は，安全であると確認できたノードが通信を希望してもポリシーによって制限することができる．

3.3 ファイアウォール制御のためのエンドノード相互間の信頼モデル

エンドノードの主導によるファイアウォール制御を実現するには，制御のための判断基準が必要である．そのため，ファイアウォール制御のためのエンドユーザ相互間の信頼モデルを提案する．本提案ではエンドユーザ同士が信頼関係にある場合，ファイアウォールを制御し，接続を許可することができると考える．これらの信頼関係は，PKI(Public Key Infrastracture)[3]によって担保する．

このモデルは，エンドユーザが通信の対向となるエンドユーザを信頼することによって成り立つ．この関係を図 1 に示す．エンドユーザ相互の信頼関係（図 1 の (A)）を確認するために，信頼関係の連鎖を用いる．

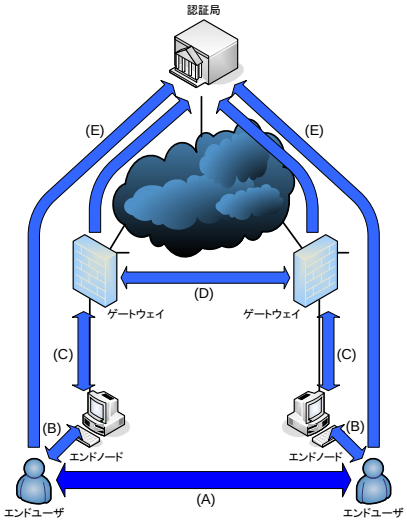

図 1: 信頼の関係

エンドノードとエンドユーザ間（図 1 の (B)）は，エンドユーザがエンドノードにログインすることによって認証が行われており，信頼関係がある．またファイアウォールの導入されたゲートウェイとエンドノード間（図 1 の (C)）は，ゲートウェイがエンドノードを自身の取り扱いとするにあたって用いた信頼関係があると考える．エンドユーザとゲートウェイは認証局を信頼し（図 1 の (E)），それによってエンドユーザ相互間の信頼関係を担保し，またゲートウェイ相互間の信頼関係（図 1 の (D)）を構築する．

ゲートウェイは，自身の取り扱いではないエンドノードについては，自身が取り扱うエンドノードの利用者が，そのエンドノードの利用者を信頼できると保証することによって信頼可能と考える．ゲートウェイが認証局を信頼すれば，認証局がエンドユーザを証明する限り，エンドユーザを個別に，事前かつ手動で設定する必要がなく，したがって P2P 型通信モデルの実現により適している．

4 提案機構の設計

本節では提案に基づいた機構の設計について述べる．

提案機構は 2 つのシステムで構成される．エンドノードに導入されるシステムをエンドノードシステムと呼び，ゲートウェイに導入されるシステムをゲートウェイシステムと呼ぶ．

4.1 エンドノードシステム

エンドノードシステムは，以下の機能を持つ．

エンドノードとエンドユーザの認証の要求

認証要求には，エンドノードの識別子として IP アドレス，エンドユーザの識別子として E-mail アドレスと公開鍵証明書が含まれる．この認証要求に，エンドユーザの秘密鍵を用いて電子署名を施し，ゲートウェイシステムに送信する．

ゲートウェイシステムに接続要求を通知

提案機構に対応するアプリケーションから対象となるエンドノードへの接続要求をゲートウェイシステムに通知する．接続要求には，接続先のエンドノードの IP アドレス，トランスポートプロトコルの種別，ポート番号が含まれる．

他のエンドノードからの接続要求の可否判断

ゲートウェイシステムから通知された，他のエンドノードからの接続要求に対して許否の判断を下す．

4.2 ゲートウェイシステム

ゲートウェイシステムは，以下の機能を持つ．

エンドノード，エンドユーザの認証

エンドノードとエンドユーザから受信した認証要求の署名を検証し，その情報をもとにエンドノードとエンドユーザの認証を行う．認証を行ったエンドノードの情報を，認証済みノードの一覧に追加する．

外部への接続要求の受け付け

認証済みのエンドノードによる外部のエンドノードへの接続要求について，許否の判断を下す．許可できる場合，接続先のエンドノードを取り扱うゲートウェイに接続要求を転送する．

外部からの接続要求の受け付け

他のゲートウェイシステムによって認証されたエンドノードによる，自身が取り扱うエンドノードへの接続要求について，許否の判断を下す．

ファイアウォールの再構成

接続要求を許可できる場合，接続が可能なようにファイアウォールを再構成する．また，エンドノードによる切断要求に従い，接続要求以前の状態にファイアウォールを再構成する．

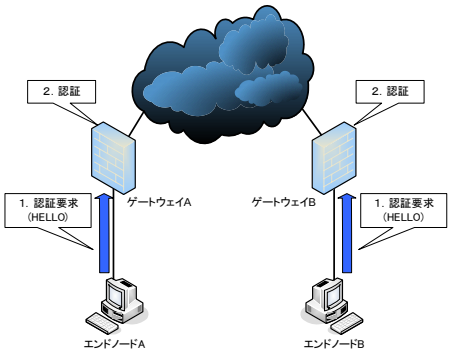

図 2: 認証要求の処理の流れ

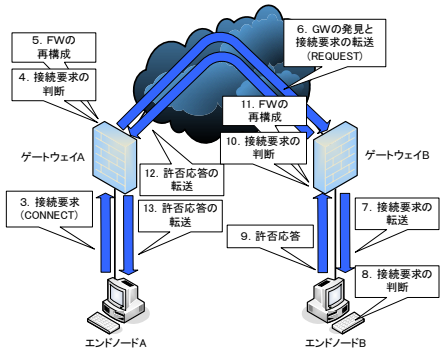

図 3: 接続要求の処理の流れ

接続要求の転送

他のゲートウェイシステムによって認証されたエンドノードによる，自身の取り扱うエンドノードへの接続要求を，当該のエンドノードへ転送する．

4.3 システム間の処理の流れ

提案機構の処理の流れについて，認証要求を図 2 で示し，接続要求を図 3 で示した．図中のエンドノード A を EN-A，エンドノード B は EN-B，ゲートウェイ A を GW-A，ゲートウェイ B を GW-B と略する．

接続要求の図では，EN-A が接続を要求する側，EN-B が接続を受け付ける側である．図中の各数字が，以下の表の各項目に対応する．

1 EN-A, EN-B によるエンドノードとエンドユーザの認証要求

2 GW-A, GW-B によるエンドノードとエンドユーザの認証

3 EN-A による，EN-B への接続要求の GW-A への送信

4 GW-A による接続要求の受け付け，および許否の判断

5 GW-A による自身のファイアウォールの動的な再構成

6 GW-A による，GW-B への接続要求の転送

7 GW-B による，EN-B への接続要求の転送

8 EN-B による，EN-A からの接続要求についての許否の判断

9 8 の応答

10 GW-B による，EN-A からの接続要求についての許否の判断

11 GW-B による自身の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

この処理によって接続が可能となり，アプリケーションは通常の接続を行うことができる．

5 プロトタイプ実装，および実験と評価

提案の有効性を確認するために，プロトタイプの実装，および実験と評価を行った．

5.1 提案機構のプロトタイプ実装

4 節で設計した提案機構のプロトタイプを実装した．実装の概要を表 1 に示す．

ゲートウェイでの接続要求の処理はマルチスレッドを用いて並列化し，複数のノードの処理を同時に処理可能としたが，ファイアウォール制御は矛盾を生じないよう直列化して処理するよう実装した．

制御に用いるファイアウォールとして，OpenBSD 3.6 付属の pf を用いた．また公開鍵証明書を扱うライブラリには，ゲートウェイシステムでは OpenSSL[6] を，エンドノードシステムでは Windows CryptoAPI[7] を用いた．

表 1: 実装環境

エンドノードシステムの実装環境	
OS	Microsoft Windows XP
ライブラリ	Windows CryptoAPI (CAPI)
言語 (コンパイラ)	C++ (MS C/C++ Compiler 13.10)

ゲートウェイシステムの実装環境	
OS	OpenBSD 3.6
ライブラリ	OpenSSL (libssl) 0.9.7d
言語 (コンパイラ)	C (gcc 2.95.3)

表 2: 実験に用いたノードの性能

ゲートウェイ PC の性能		
	GW-A	GW-B
CPU	Pentium II 300MHz	Pentium II 400MHz
RAM	256MB	128MB
HDD	IBM-DHEA-38451	IBM DTTA-351010
OS	OpenBSD 3.6	OpenBSD 3.6

エンドノード PC の性能		
	EN-A	EN-B
CPU	Pentium III 500MHz	Pentium 4 2.53GHz
RAM	128MB	1024MB
HDD	Seagate ST36421A	Seagate ST380011A
OS	Windows XP SP2	Windows XP SP2

5.2 実験に用いたノードと実験ネットワーク

実験には 4 台の PC を用いた．うち 2 台がゲートウェイシステムを導入するゲートウェイで，それぞれ GW-A，GW-B とする．残りの 2 台がエンドノードシステムを導入するエンドノードで，それぞれ EN-A，EN-B とする．各 PC の性能を表 2 に示す．

この各 PC 間を図 4 のとおりに接続した．各 PC 間は 100Mbps で接続されている．

5.3 評価項目と実験による評価および考察

実験では，接続に要する時間を計測し，通常の接続と比較した際のオーバーヘッドとして各処理にかかる時間を求め，ボトルネックとなる部分の調査を行った．その結果を，受信側である GW-A について表 3 に，送信側である GW-B について表 4 に，それぞれ示す．

表 4 の項目 (E) には，受信側のゲートウェイである GW-A のすべての処理が含まれている．項目 (A) に要する時間が大きいが，これは通信処理のオーバーヘッ

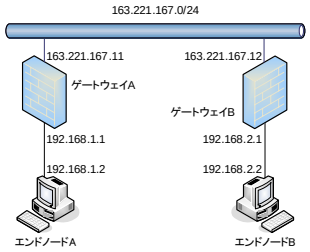

図 4: 実験ネットワークのトポロジ図

表 3: 測定結果: GW-A における各部の処理時間

処理部位	[ミリ秒]
(A) エンドノードへの要求転送	160.613
(B) ファイアウォールの再構成	51.659
(C) 接続要求メッセージの検証	30.520
(D) 通信メッセージの解釈	0.617

ドであり，実装上の問題であると考える．項目 (B), (C), (F), (G), (H) は，公開鍵証明書の処理やファイアウォールの再構成など提案機構の設計上必要な処理である．また，項目 (I) も設計上必要な処理であるが，他の項目との違いは，今回の計測は小規模な実験環境で行ったため値が小さいが，現在の手法では接続先ゲートウェイまでの往復時間 (RTT) に依存することである．実際のインターネット環境ではより大きくなると考えられる．

提案機構を用いた場合，通常の接続の手順に比べ処理に時間を要するが，この時間は項目 (I) 以外の処理ではほぼ一定である．本提案機構では接続後に必要な処理はないため，一度接続を行った後に長時間接続を維持する P2P 型のアプリケーションを設計することで，遅延の影響を低く抑えられると考える．

表 4: 測定結果: GW-B における各部の処理時間

処理部位	[ミリ秒]
(E) 接続要求の転送	248.049
(F) 通信メッセージの署名	55.919
(G) ファイアウォールの再構成	28.421
(H) 接続要求メッセージの生成	11.679
(I) 接続先ゲートウェイ検出	2.555

6 まとめと今後の課題

本研究では，エンドノード主導のファイアウォール制御機構を提案した．また提案に基づいた機構を設計し，プロトタイプ実装を行った．この実装を用いて，提案機構についてのオーバーヘッドを計測し，提案や設計に起因するオーバーヘッドと実装による問題を切り分け，明らかにした．

今後の課題としては，以下が挙げられる．

- よりオーバーヘッドの小さな，相手先ノードを管轄するゲートウェイの検出手法の考案
- 接続要求の許否判断のポリシー定義の手法
  本実装では接続要求の許否判断のポリシーは単純なものを用いたが，いかなるポリシー記述が可能なら過不足内判断が可能であるかについては議論の余地がある
- エンドノードの安全性を確認するための検疫システムとの密接な連携の仕組み

参考文献

[1] M. Leech, Y. Lee, R. Kuris, D. Koblas, L. Jones, “SOCKS Protocol Version 5,” RFC 1928, 1996.

[2] S. Lodin, C. Schuba, “Firewalls Fend Off Invasions from the Net,” IEEE Spectrum 35 (2), 1998.

[3] C. Adams, S. Lloyd: “Understanding Public-Key Infrastracture: Concepts, Standards, and Deployment Considerations,” Mamillan Technical Publishing, 1999.

[4] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, B. Palter, “Layer Two Tunneling Protocol,” RFC 2661, 1999.

[5] B. Patel, B. Aboba, W. Dixon, G. Zorn, S. Booth: Securing L2TP using IPsec, RFC 3193, 2001.

[6] OpenSSL Project, `http://www.openssl.org/`.

[7] Microsoft Corporation, CryptoAPI System Architecture, `http://msdn.microsoft.com/library/en-us/security/security/cryptoapi_system_architecture.asp`.