平成20年度卒業論文
論文題目

分散ファイアウォールの連携

神奈川大学　工学部　電気電子情報工学科　木下研究室
学籍番号200502780
国崎　大地

指導担当者　　木下　宏揚　教授

目 次

1 章 序論

現在、生活の一部としてネットワークは欠かせないものになった。誰でも容易にインターネットを使えるようになった反面、その中で、企業などのネットワークでは、インターネットなどの外部ネットワークを通じて第三者が侵入し、データやプログラムの盗み見・改ざん・破壊などが行なわれる被害が出ている。この様な事のないように、外部との境界を流れるデータを監視し、不正なアクセスを検出・遮断する必要がある。このような機能を実現するシステムがファイアウォールである。[1,4] ファイアウォールの目的は、必要な通信のみを通過させ、不要な通信を遮断することであり、通常内部のネットワークから外部はアクセスできるが、外部から内部のネットワークにアクセスができないような制御が一般的である。[2]

ファイアウォールの設置方法として、外部ネットワークと内部ネットワークの境界のみに設置する方法（集中型）と、境界のみでなく内部ネットワークの部署単位でも設置する方法（分散型）が考えられる。[3] 集中型のファイアウォールでは一箇所で制御しているので管理はしやすいが、部署ごとの異なるセキュリティポリシーに対応できない、内部からの攻撃に対処できないなどの問題点がある。そのため、さらなる強固なセキュリティを実現するために注目されたのが分散型ファイアウォールである。分散型のファイアウォールなら上記の問題も解決できる。さらに、部署ごとに異なるセキュリティポリシーを設定できるので柔軟で強固なセキュリティを実現できる。[9]

本稿では、仮想マシンを用いたネットワークを提案し、分散ファイアウォールのセキュリティポリシー（ルール）をほかのファイアウォールと連携させることにより、全体としてのルール数を減らし、負荷の要因となるマッチング回数軽減によるファイアウォールの最適化（処理能力向上）について実装実験を元に考察する。

2 章 基礎知識

2.1 ファイアウォール

ファイアウォールとは、インターネットなどの信頼できないネットワークからの攻撃や、不正アクセスから組織内部のネットワークを保護するためのシステムである。[6] ファイアウォールは、ネットワークとネットワークを分離し、特定のプロトコルや宛先パケットしか通過させないパケットフィルタリングと、インターネット上のサーバに代わって内部のネットワークにサービスを提供するプロキシサーバの機能により、内部ネットワークと外部ネットワーク間のパケット転送を行う。[8] 実現方式として、パケットレベルでアプリケーションの使用するポートを制御するパケットフィルタリング、およびアプリケーション（プロキシによる代理応答）で内外のネットワークを通信するアプリケーション・ゲートウェイ方式がある。[17]

	パケットフィルタリング	ステートフルパケットフィルタリング	アプリケーションゲートウェイ
制御	Ｉｐパケットのヘッダ情報（アドレス、ポート番号など）といった静的ルールで判断	Ipパケットのヘッダ情報だけでなく、時間や履歴といった動的ルールで判断	Ｉｐアドレス、ポート番号、アプリケーションレベルのデータで判断
処理速度	高速	中速	低速
CPU／メモリ資源	少ない	中程度	多い
安全性	低い	中程度	高い

図 2.1: 各方式の特徴

2.2 ファイアウォールの設置方法

ファイアウォールの設置方法には、以下の二通りが考えられる。

・外部ネットワークと内部ネットワークの境界のみに設置する方法（集中型）

・外部ネットワークと内部ネットワーク境界のみでなく、内部ネットワークの部署単位でも設置する方法（分散型）

上記二通りについて利点および問題点について述べる。

2.2.1 集中型ファイアウォール

集中型ファイアウォールでは外部ネットワークと内部ネットワークの境界のみに設置するため以下のような利点問題点が考えられる。[19]

利点

・管理が一元化できる。

問題点

・ 部署ごとの異なるセキュリティポリシーに対応できない。

・内部からの攻撃に対処できない。

・ウイルスに感染したパソコンの持込による被害拡大に対処出来ない。

集中型ファイアウォールの例を図に示す。

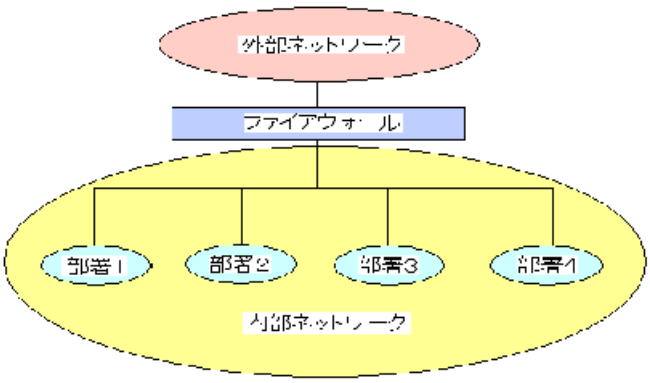

図 2.2: 集中型ファイアウォール

2.2.2 分散型ファイアウォール

集中型ファイアウォールに対して分散型ファイアウォールでは内部ネットワークの部署単位でもファイアウォールを設置するので以下の利点が考えられる。

利点

・ 部署ごとの異なるセキュリティポリシーが設定可能。

・外部からの攻撃だけでなく、内部からの攻撃に対する防御が出来る。

・ウイルスに感染したパソコンの持込による被害拡大の防止。

分散型では上記のような利点があるため、集中型よりも強固で柔軟なセキュリティを実現できる。しかしながら、分散型でも集中型でも外部ネットワークと内部ネットワークの境界に設置されたファイアウォールの負荷はかわらない。負荷の増加による通信速度の遅延は内部ネットワーク内のファイアウォールにも影響を及ぼすので好ましくない。[20,21]

分散型ファイアウォールの例を図に示す。

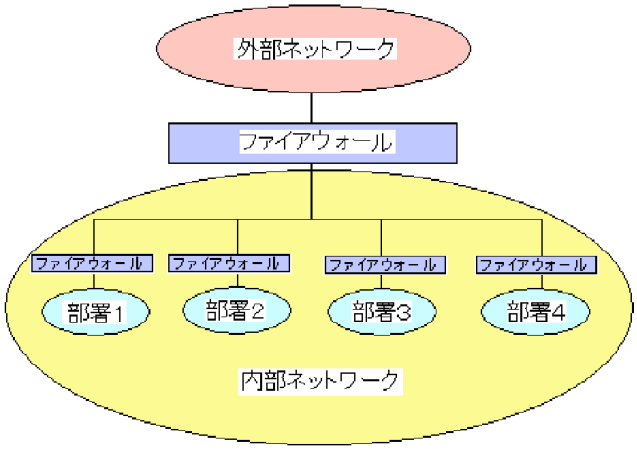

図 2.3: 分散型ファイアウォール

2.3 パケットフィルタリング

ルータの機能とは、パケットの最終目的地をもとに複数のネットワークインターフェイスの間でパケットを転送することである。通常パケットの転送はカーネルの機能である。パケットフィルタリングは転送の際に特定のパケットのみ通過させ、それ以外のパケットは転送を阻止することによって内部ネットワークを保護する。通過させるかどうかの判断基準をルールリストと呼ぶ。送られて来たパケットの IP ヘッダ情報を元に通信を許可するか、または拒否するかをルールとのマッチングを行い判断する。[5,14]

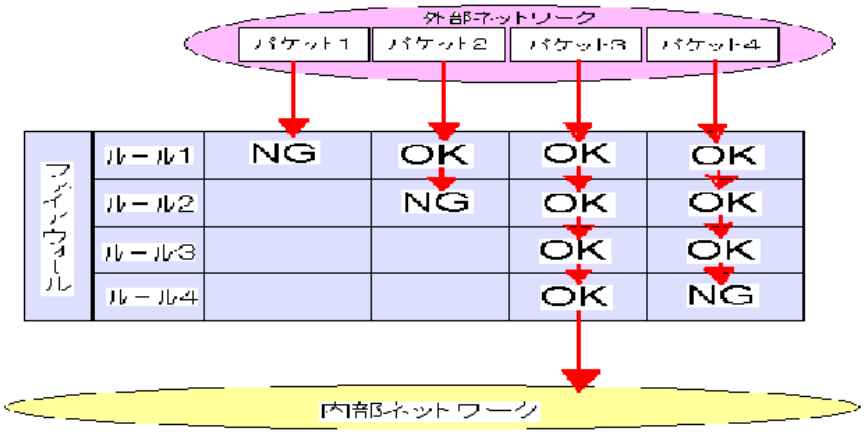

図 2.4: パケットフィルタリング

2.3.1 IPFW

ルータにIPFWルールの設定を行うことでファイアウォールを実現できる。IPFWフィルタリングではパケットごとに、マッチするルールが見つかるまでルールリストを調べる。IPFWのルールの例を図に示す。[12,13,15,16]

```
add pass tcp from any to 133.72.88.10 22
add deny udp from any to any
```

図 2.5: IPFWのルールの例

上記ルールの各コマンドは以下の役割を持つ。

●操作

・add：ルールを追加する

・delete：ルールを削除する

●処理方法

・pass：パケットを通過させる

・deny：パケットを破棄する

・count：通過パケットのカウントをする、制御は行わない

●プロトコル

・tcp：tcpプロトコルのみ制御する

・udp：udpプロトコルのみ制御する

・icmp：icmpプロトコルのみ制御する

・ip：すべてのプロトコルにおいて制御する

2.4 プロキシサーバ

パケットフィルタリングが、カーネルのパケット転送機能により適切なパケットのみを通過させるのに対して、プロキシサーバではカーネルのパケット転送を禁止し、サービスごとのデーモンを置き換えることにより、パケット転送を行う。例えば、WWW のプロキシサーバではクライアントがファイアウォール外の WWW サーバにアクセスする際、プロキシサーバがクライアントに代わって WWW サーバからデータを取得し、それをクライアントに転送する。一般にプロキシサーバは、通常ホストより強固な認証手段を実装している。

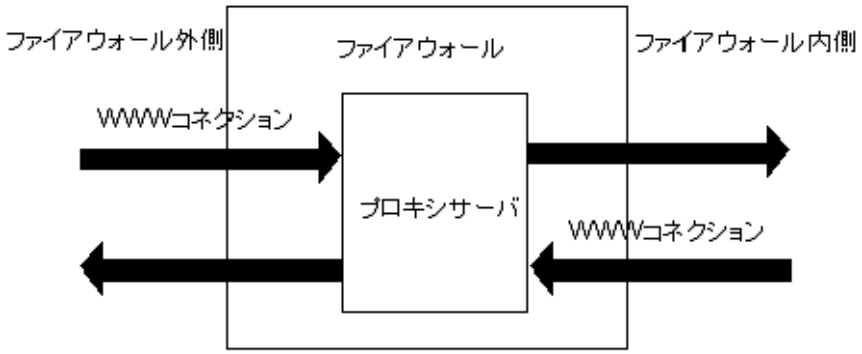

図 2.6: プロキシサーバ

2.5 仮想マシン

今回はQEMUというエミュレータを用いて仮想化を実現した。仮想化とは詳しくはプラットフォーム仮想化のことで、ハードウェアプラットフォーム上でホストプログラム（制御プログラム）が擬似的なコンピュータ環境を生成し、ゲストソフトウェアに対して「仮想機械」を提供するものである。ゲストソフトウェアは、それ自体もオペレーティングシステムであるのが一般的で、あたかも独立したハードウェアプラットフォームにインストールされたかのように動作する。[7] 単一の物理マシン上で複数の仮想機械をシミュレート可能なことが多く、仮想機械の個数はホストであるハードウェアリソースによって制限される。ゲストOSとホストOSは一般に同一である必要はない。ゲストシステムは特定の周辺機器（ハードディスクドライブやネットワークカード）へのアクセスを必要とすることが多く、その場合その機器とゲストのインタフェースを提供する必要がある。プラットフォーム仮想化の手法はいくつか存在する。[10]

その中のひとつがエミュレータを用いた仮想化である。エミュレータとは仮想機械によってハードウェア全体を擬似的に再現することで、全く異なるアーキテクチャのハードウェア向けのゲストOSを修正することなしに動作させることができる。これは、新たなCPUなどのハードウェア開発が完了する前にソフトウェアを並行して開発する手法として使われてきた。具体例としては、Bochs、PearPC、Virtual PCのPowerPC版、QEMU、Hercules emulator（IBMのメインフレームのエミュレータ）などがある。エミュレーションのための技法は様々で、有限オートマトンを使った技法から、仮想化プラットフォーム上での動的再コンパイル技法まである。 [11]

2.6 ファイアウォールの処理能力に影響を及ぼす負荷の要因と改善方法

ファイアウォールの処理能力に影響を及ぼす負荷の要因として、受信パケットのサイズ、送られて来たパケットと設定したルールとのマッチング処理、経路情報の増加、などが考えられる。ファイアウォールのメモリの量や処理能力には限界がある。従って、負荷の増加は通信速度の低下を招き、その環境下にある部署にも影響を及ぼす。ルール数が増えるとマッチングを行う際に負荷がかかりスループットに影響を与える。すなわち、「ルール数を減らす」＝「負荷の軽減」と言える。[18]

よって負荷を軽減させる方法として以下の事が考えられる。

・操作とプロトコルが同じルールが連続になるように並び替える。

・ルールの統合を行う。

・ファイアウォール間でのルールの受け渡し、受け取ったルールの並び替え。

2.7 最適化

最適化と言っても、目的に応じて様々な手法がある。その中でも本稿では、最適なフィルタリングについて考える。ファイアウォールにかかる負荷要因として送られて来たパケットと設定したルールとのマッチング処理が考えられる。大きい行番号でマッチング処理が行われるほどマッチング回数が増加して、ファイアウォールにかかる負荷は増加する。従って、マッチング処理回数が最小となるフィルタリングを最適と定義できる。

マッチングは低い行番号から順に行われるため、設定ルール数の削減、および使用頻度の多いルールを低い行番号に設置することでマッチング回数を削減できる。従って、出現頻度の高いパケットは低い行番号の場所、出現頻度の低いパケットは高い行番号の場所でマッチングを行わせることで負荷の軽減が見込める。[22]

2.7.1 Quine-McClaskey の方法

Quine-McClaskey の方法 [22] は与えられた論理関数の積和形論理式の最小化を求める方法である。論理式の加法標準形を IP アドレスと考えるとこの手法を用いることで IP アドレスを基にルール数を削減できる。

Quine-McClaskey の方法の手順を示す。

１．論理式を加法標準形で表す。

２．各最小項（IP アドレス）を 2 進数表示の 1 の数によってグループに分ける。1 の数の同じ項を同一グループとし、小さい順に並べる。

３．１の数がひとつだけ違うグループの項同士を比較し、差が 2 のべき乗の項を取り出す。

４．取り出した項同士で再び比較を行う。（比較できなくなった項は主項となる）

５．すべての項が主項となるまで繰り返す。

６．縦軸に主項、横軸に最小項を記した表を作成する。

７．各最小項を少なくとも一回は含むような主項の組み合わせを求める。

3 章 提案システム

3.1 提案システム

ファイアウォールの処理能力に影響を及ぼす負荷の要因として、受信パケットのサイズ、送られて来たパケットと設定したルールとのマッチング処理、経路情報の増加、などが考えられる。本稿では、ルールとのマッチング処理に注目した。各ファイアウォールに設定するルール数を減らすことにより、ルールのマッチングの最適化に努め、ファイアウォールのメモリ使用量の軽減、パケットが通過する際の遅延時間減少が見込める。分散環境下にあるファイアウォールとその上にあるファイアウォールとの間でルールの受け渡しをすることによって、各ファイアウォールでのルールのマッチング回数を減少させる事が出来る。ルール数とスループットは反比例の関係にあるため、ルール数が減少することで、各ファイアウォールのマッチング回数が減り、負荷の軽減につながる。

ルール統合の手順

部署ごとのセキュリティポリシーを集める。

集めたセキュリティポリシーに基づいてルールの統合し最適化を行う。

最適化の論理式に基づいて各ファイアウォールにルールを返し、設定する。

部署ごとのセキュリティポリシーに変更があった場合は ～ をやり直す。

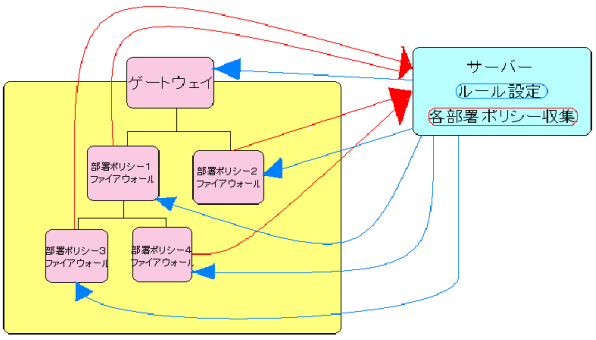

図 3.1: システム概要

3.2 システム詳細

本研究では3.1で述べたような手順が自動的に行われるようプログラムに書いて実装する。まず、分散環境下にあるファイアウォールにそれぞれのセキュリティポリシー（ルール）を設定する。今回はiptablesに直接、プロトコルやポート番号に応じてどのようなアクションをとるか記入した。次にセキュリティポリシーを自動で回収できるようにさせるために、すべてのファイアウォールに共通の鍵を持たせて認証なしでアクセス出来る様にした。このような流れで回収した各ファイアウォールのセキュリティポリシーをプログラムに読み込ませる。このプログラムでは、プロトコルやポート番号やアクションの違いに応じてルールを並び替える動作を行う。これによってプロトコルやポート番号やアクションからなるルールが似たものから順番に並び替わることになる。並び替わったルールの上下を比べることで、同じルール、同意義のルールを見つけられる。ここでルール統合の処理を行ってルール数自体を減らす。統合されたルールと統合されなかったルールを最適化の論理式に基づいて各ファイアウォールに返し、設定する。上記のようにして分散ファイアウォールの連携を実装する。

4 章 実装実験

３章で述べた方法で実装実験を行った。ここでは実装実験で行った内容を簡易的に述べる。

4.1 実装例１

下図のようにファイアウォールを A、B、C、D の 4 台を用意し 3 台を並列に、もう１台を３台に対して縦列に接続することで分散型ファイアウォールを実現する。

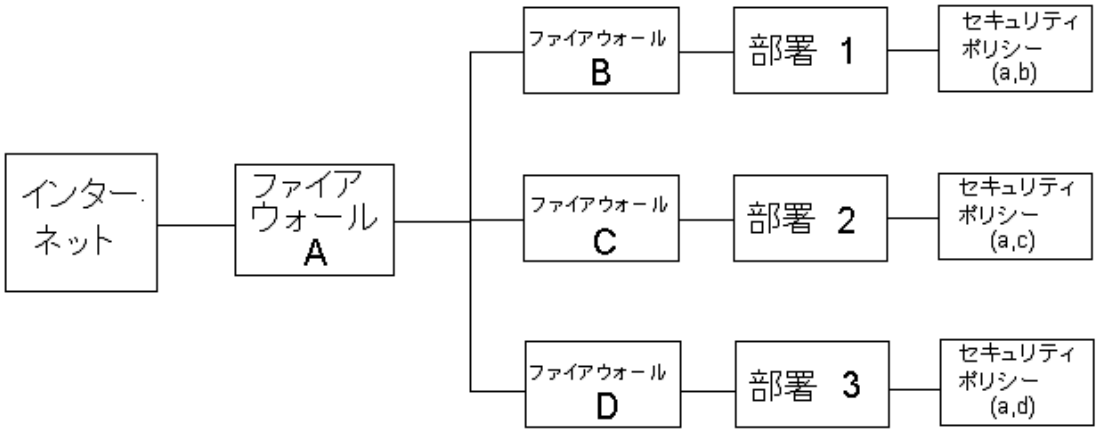

図 4.1: 実装モデル１

例えば、セキュリティポリシーがa、b、c、dという４種類存在するとする。図4.1のように、ファイアウォールB、C、D にそれぞれ(a,b)、(a,c)、(a,d) という形でセキュリティポリシーが設定されていたとすれば、ファイアウォールB、C、D はどれも同じaというセキュリティポリシーを持つことになる。このaというセキュリティポリシーをファイアウォールAに設定すれば、ファイアウォールB、C、Dはわざわざ同じ動作を３台別々に行う必要がなくなる。個々のファイアウォールの負担を減らすことでシステム全体としての負荷の軽減が実現される。

下図はセキュリティポリシーaを統合してファイアウォールAに設定した場合の図になる。

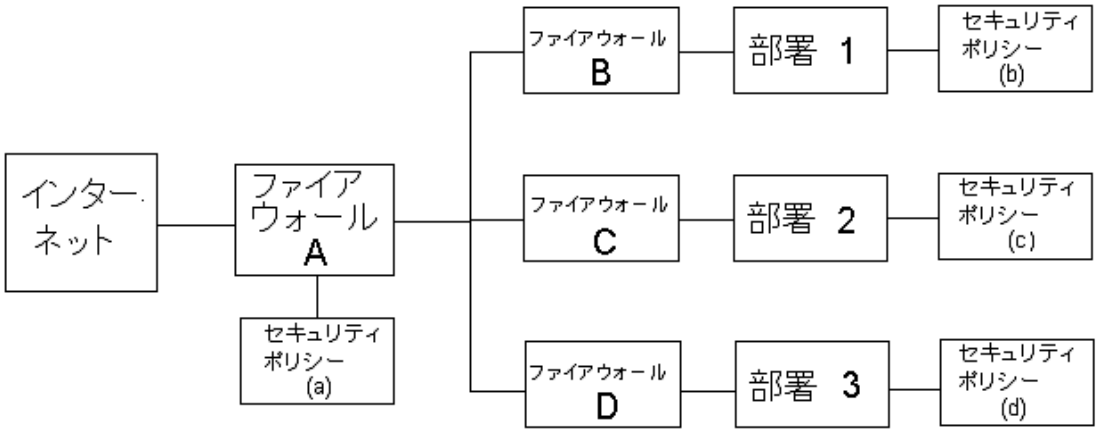

図 4.2: 実装モデル１（最適化後）

4.2 実装例２

ここで、さらにセキュリティポリシーを増やした場合について考えてみる。a〜m までの計 13 個のセキュリティポリシーが下図のように各ファイアウォールに配置されていたとする。

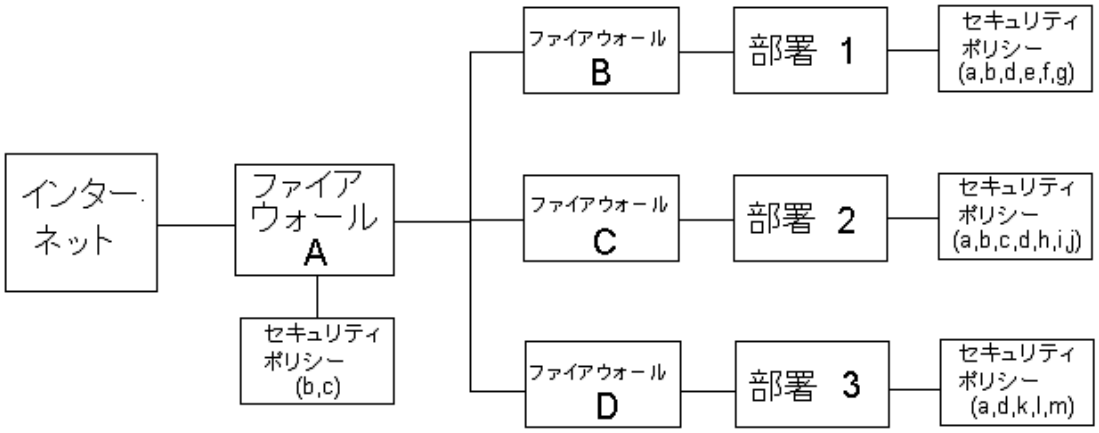

図 4.3: 実装モデル２

図4.3の場合だと、インターネットから入って来たパケットがファイアウォールA を通過して、ファイアウォールBを通過して部署１に届くまで計８個のセキュリティポリシーとのマッチングを行うことになる。同様に部署２まで届くには計９個、部署３に届くまでは７個となる。これをファイアウォール同士を連携させて、ルール統合、最適化を行うと、部署１，２，３どれも計７個のセキュリティポリシーとのマッチングで済む。このようにしてルールを統合させることによってシステム全体としてのルールとのマッチング回数を減らし、分散ファイアウォールの連携による負荷軽減を実現した。

実装モデル２について最適化したものを下図に示す。

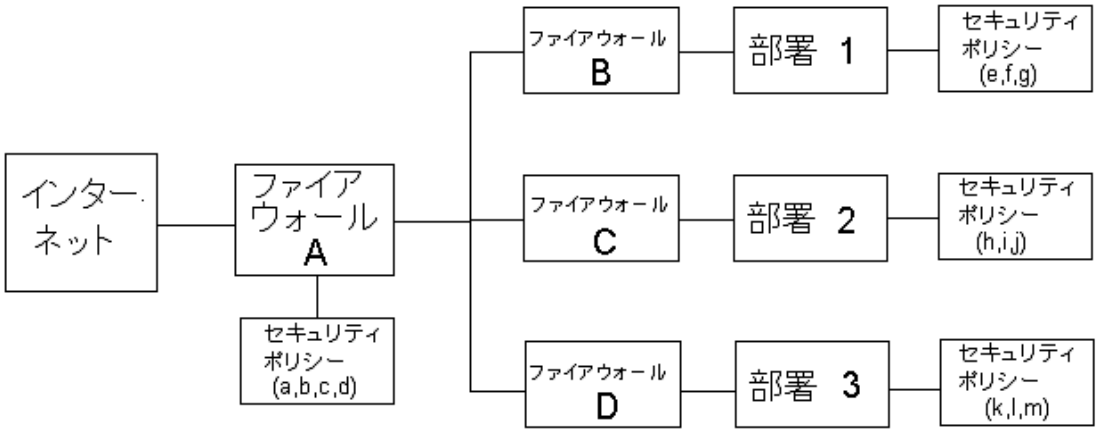

図 4.4: 実装モデル２（最適化後）

5 章 結論

本研究では、分散型ファイアウォールの連携による最適化、すなわち負荷の軽減をルールとのマッチング処理に注目して仮想マシンを用いた実装実験にて実現した。今回は仮想マシン上での実験ではあったが、動作確認も出来た。また、実際のネットワーク上での動作では、さらに複雑なネットワークになるため、セキュリティポリシーの数も増し、あらゆる攻撃に対して制御できるようにしなければならない。今後は、特定のセキュリティポリシーだけでなく、実際のネットワークでも対応できるような様々なセキュリティポリシーを持った分散型のファイアウォールでも、ファイアウォールの連携による負荷の軽減が見込めるようなシステムの検討が必要である。

謝辞

本研究を進めるにあたって、終始熱心に御指導して頂きました木下宏揚教授に感謝致します。また、円滑に研究を進めるためにお世話になった木下研究室助手鈴木一弘氏に感謝致します。最後に神奈川大学木下研究室の諸兄にも感謝致します。

2009年2月
国崎　大地

参考文献

[1] IT用語辞典:http://kaden.yahoo.co.jp/dict/?type=detailid=1397

[2] @IT：http://www.atmarkit.co.jp/aig/02security/firewall.html

[3] Cosminexus　システム設計ガイド：
http://www.hitachi.co.jp/Prod/comp/soft1/
manual/pc/d3M0420/EM040101.HTM

[4] 情報処理推進機構：http://www.ipa.go.jp/security/

[5] 「iptables」によるパケットフィルタリング：
http://www.crimson-snow.net/hmsvr/
centos/memo/iptables.html

[6] くわぞうメモ:http://www.kuwazou.net/blog/2006/05/iptable.html

[7] UNIX の基本操作：http://web.kanazawa-u.ac.jp/univ/center/ma5-6.html

[8] TECH WORLD：http://www.techworld.jp/

[9] ためになるホームページ：http://www.booran.com/

[10] シェルスクリプト入門：
http://www.k4.dion.ne.jp/ mms/unix/shellscript/index.html

[11] Hot　Linux：http://www.hot-linux.org/redhat/?rec$_n o = 7$

[12] さかお　まい：C言語入門 株式会社エクスメディア（2005）

[13] 山田　和夫：基礎からのC ソフトバンククリエイティブ株式会社（2008）

[14] 久米　原栄：IP ルーティング入門 ソフトバンククリエイティブ株式会社（2007）

[15] デイブ　テイラー：unix 入門下 株式会社ピアソンエデュケーション（1999）

[16] 小野　斉大：unix コマンドブック ソフトバンクパブリッシング株式会社（2003）

[17] 木下　宏揚：情報処理工学 コロナ社（2001）

[18] 宮地　伸晴：分散ファイアウォールにおけるルールの配置と処理の最適化（2007）

[19] 宝木　和夫：ファイウォール-インターネット関連技術について 昭晃堂（1999）

[20] 下條　敏夫：ポリ氏による複数ファイアウォールの一括設定方式の提案と実装 信学論（2004）

[21] 油川　良太：分散型ネットワークモニタリングによる不正アクセス早期発見システム 信学論（2003）

[22] 田丸　啓吉：論理回路の基礎　工学図書（1989）