愛知工業大学研究報告
第33号B　平成10年

HTTPプロクシを利用した代理サーバの設計

Making a proxy server using HTTP proxy

杉浦 敦也 †,　　羽賀 隆洋 ‡

Atsuya SUGIURA,　Takahiro HAGA

Abstract　Today, many organization and person are offering service by way of Internet or other network, and using some security system to save their system from what they don't want. But, almost security system used in today restrict to user to save the system. Usually, it's restrict to service that user usually don't use. But, that service is sometime necessary. Now, I pick up a firewall,that's one of the such security system. Usually, they use proxy server to communicate beyond the firewall. But,they need permission to drive that on firewall host. There is case of limit to give a account or to ban to execute daemon process, to improv security level. Almost administrator offers proxy service for well known service such like WWW. But, They can't communicate using other service. In this paper, I make proxy server drive communicating each other using HTTP proxy running at Aichi Institute of Technology, and examine it.

1　はじめに

今日，多くの団体や個人では，インターネットやその他のネットワーク等を経由して様々なサービスの提供をし，同時に，自分のシステムを他人による自分の望ましくない行為から守るためにセキュリティシステムを実装している．しかしながら，今日実装されているセキュリティシステムの殆んどがユーザに制限を加える事によって実現されている．通常，ユーザが必要とする可能性の低いサービスに対して制限を加えるわけだが，必要とする可能性が無い訳ではなければそれが重大な問題となる．

今回そんなセキュリティシステムの一つであるファイアウォールを取り上げる訳だが，ファイアウォールを超えて通信する為に，代理サーバを経由して通信することは前々から行なわれてきた．しかしながら，代理サーバはファイアウォール上にこれを動かす権限が必要である．ファイアウォール上のホストはセキュリティレベルの向上の為，ログインする権限をごく限られたユーザにしか与えなかったり，そこで代理サーバのような常駐型のプログラムを起動することを禁止していたりする場合がある．WWWのような広く使われているサービスにたいしては，多くの管理者がこれを経由する為の代理サーバを用意しているが，それ以外の通信をする事はできない．

本論文では，愛知工業大学で実装されているHTTPプロクシを利用して，情報を相互交換し駆動する代理サーバを設計し，それについて検討してみた．

2　ファイアウォールとは

プログラムは常にそれを使う人間の思惑通り動くとは限らない．プログラムにバグがある場合や，設定を誤ってる場合，あるいは，それを動作させるために必要なOSやドライバ等のバグによって誤動作する場合等がある．特に大規模なプログラムでは，そのサイズから想像する以上にバグが多いと言われている．ネットワークやセキュリティ等のプログラムのバグは，重大なセキュリティホールを作る場合も多く非常に危険である．

ところが，如何なるプログラムであろうとも実行しなければバグが多いかどうかは問題にならない．即ち，プログラムを実行しなければセキュリティホールがあるかどうか問題にならないのである．その為に外部と接触するマシンでは最小限のプログラムを起動し，それ以外のプログラムは外部との接触が出来ないマシンで起動する事によって，セキュリティホールを最小限に抑えることが出来る．

その為にも，ネットワークの外部と内部の通信を制限するファイアウォールが重要となる．

†愛知工業大学大学院 学生 (豊田市)
‡愛知工業大学 情報通信工学科 (豊田市)

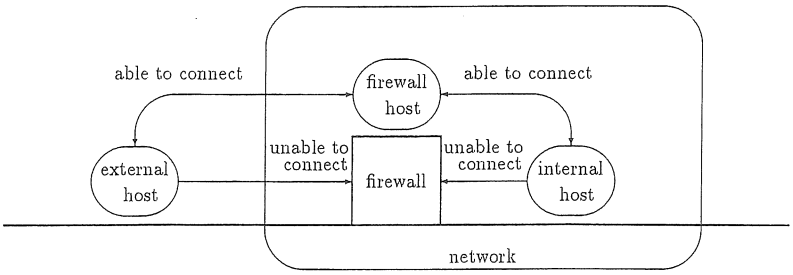


図 1: ファイアウォールの構造

2.1 ファイアウォールの動作

ファイアウォールを実装しているネットワークの場合，外部のホストからファイアウォールホストへの接続は出来るが，それ以外のホストへ接続は出来ない．ネットワークの内側のホストについては，ファイヤウォールホストの場合は外部へも接続できるが，それ以外のホストから外部のホストへ接続することは出来ない (図1参照)．

実際には，すべての通信を阻止するのではなく，特定の TCP ポートへの接続を阻止する事が出来，ウェルノンポートのみの制限である事が多いのだが，愛知工業大学のファイアウォールは全てのポートを完全に閉鎖している．

2.2 ファイアウォールの問題点

ファイアウォールは，ネットワークの内部と外部の任意の通信方式，経路を遮断する壁となる．当然，それによって外部からの悪意を持った侵入を防ぐわけだが，当然悪意の無い侵入，あるいは内部から外部への接続さえも制限してしまうのである．通常，よく必要とされる通信は外部と内部を中継する代理サーバを経由したり，IP フィルタリングを行う等の方法で解決される場合が多いが，それ以外の通信路は完全に遮断されてしまう．

実際に代理サーバ等を立ち上げる為には，ファイアウォール上のホストを扱う権限を持っていなければならず，また，そういったホストでの常駐プログラム等の起動を禁じられてない事が必要となる．この条件に沿わなければ，ファイアウォールを越えた接続が必要となった時であっても，そこで代理サーバ等を立ち上げて接続するといった手段を用いる事が出来ないのである．

3 HTTP プロクシの利用

今回設計したプロクシは，ファイアウォール上のホストに代理サーバを立ち上げられる権限の無い場合に対応するために，多くのネットワークで駆動している HTTP プロクシを経由して行うもので，これによってファイアウォールを超えて通信する事が出来る．

3.1 HTTP とは

HTTP とは，Hypertext Transfer Protocol の略で，WWW 資源の転送等に一般的に使われているアプリケーション層のプロトコルである．これは，リクエストとそれに対するレスポンスという枠組で行われる．リクエストは，メソッド (今回は常に POST メソッド)，URI，プロトコルバージョン (今回は HTTP/1.0 を使用) からなるリクエストライン，ヘッダーフィールド (今回は Content-type 及び，Content-length のみを指定)，時にオブジェクトボディ (今回は POST メソッドを使用するため常に存在する) からなり，レスポンスは，ヘッダーフィールド (今回は Content-type のみ)，オブジェクトボディからなる．

HTTP は基本的に，クライアントからサーバへ要求を出し，それが終ってから，サーバからクライアントへ返答すると言ったように，片方向づつの通信となる．

3.2 HTTP プロクシとは

今回ファイアウォールを越えるために使ったのは，HTTP を中継する代理サーバで HTTP プロクシと呼ばれるものである．

ウェブブラウザ等，HTTP クライアントは通常ユーザが URI で指定したサーバに接続し，その資源を転送する要求を出し，サーバはその資源を転送する．HTTP プロクシを用いる場合，クライアントは HTTP プロクシに接続して，ユーザが指定した URI の転送要求を出す．HTTP プロクシは要求されたサーバへ接続しサーバから得られた資源をクライアントに送信する．これはファイアウォール内から外部の WWW ページを参照したい場合等に使われる．

もう一つ，愛知工業大学では外部からファイアウォール内のサーバを代理接続する代理サーバが駆動している．これは、外部のネットワークからは HTTP サー

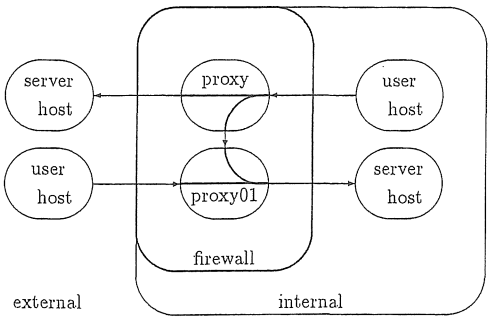


図 2: 駆動しているプロクシ

バであるかのよう振るまい，要求されたURIからサーバを判別し，該当するサーバへ代理接続する．ここで注意するのは，URIに記されたサーバではないという事である．クライアントにはURIに記されたサーバがHTTPサーバであると見せかけるので，URIに記されたサーバはこの代理サーバであり，URIは短縮表現で要求される．

愛知工業大学で，前者はSquid，後者はDeleGate，と呼ばれるソフトウェアが使われている為，以降Squid，DeleGateと呼ぶ．

3.3 URIの短縮表現

今回ではHTTPのみ扱っているのでURIは以下のような形で表される．

```
"http://" <host> [":" <TCP-port>] [ <path> ["?" <searchpart>]]
```

サーバがSquidのようなHTTPプロクシ(DeleGateのようにHTTPサーバであるかのように振舞うものを除く)でない場合，プロトコルとアドレスは自明である為，例えば，

```
http://www.aitech.ac.jp/wing/~atsuya/
```

であれば，以下のように短縮表現がなされる．

```
/wing/~atsuya/
```

これは先に説明したDeleGateへのURIである．DeleGateはこの場合，wingをいうキーワードによって，羽賀賀研究室で駆動しているサーバと判断する．

3.4 環境

愛知工業大学の環境では，内部のユーザが外部のサーバに接続するためのSquid (ホスト名 proxy) と，外部のユーザが内部のサーバに接続するためのDeleGate (ホスト名 proxy01) が，起動している．(図2参照)．

この結果内部のホストから内部のサーバへ接続する場合，内部のホストからproxyのSquidに接続し，そこからproxy01のDeleGateに接続し，そこから内部のサーバへ接続するという経路をとる事になる．

ファイアウォール上のホストでは常駐プログラムは禁止されているので，ここで代理サーバを起動することは出来ない．

4 代理サーバの実装

今回設計した代理サーバでは，まず，ファイアウォールの内部と外部で通信すべきプロセスを走らせ接続待機させる．

外部から内部への送信は内部で駆動しているプロセスからSquidに接続し外部で駆動しているプロセスへ接続する為のURIを指定すると，HTTPプロクシは外部で駆動しているプロセスに接続しリクエストメッセージを送信する．そのメッセージを受信した外部で駆動しているプロセスはそれに対するレスポンスメッセージのオブジェクトボディを用いて単方向通信経路が確立される．

内部から外部の通信は，外部で駆動しているプロセスからDeleGeteを経由してHTTPサーバへ接続し内部で駆動しているプロセスに接続する為のCGIプログラムのURIを指定する．CGIプログラムは内部で駆動しているプロセスに接続し，そのCGIプログラムからHTTPサーバを通して外部で駆動しているプロセスへ単方向通信経路が確立される．内部で駆動しているプロセスからCGIプログラムへは双方向で通信できるが，外部で駆動しているプロセスからCGIプログラムへの送信は出来ないので，内部から外部への単方向通信となる．

HTTPは要求を送ってから返答を返す方式なので，リアルタイムに双方向通信を行うことが出来ない．特にリクエストメッセージのオブジェクトボディは，全て代理サーバへの送信を全て終えてからサーバへリクエストメッセージを送る為，これを用いてリアルタイム通信は出来ない．双方向通信を要する場合は2つの接続を要し，それぞれのレスポンスメッセージのオブジェクトボディを利用する．

4.1 サーバの動作

今回作成した代理サーバは以下のように動作する．まず，内部サーバ起動用URIを指定して外部サーバを起動すると内部サーバが起動し接続する．

本システムは外部から操作する事を目的とし，外部サーバへ制御命令として，接続方向 (内部から外部への接続か，外部から内部への接続か)，待機するポート番号，接続するサーバのアドレス，接続するポート番号，を入力するとその中継サービスを開始して，ク

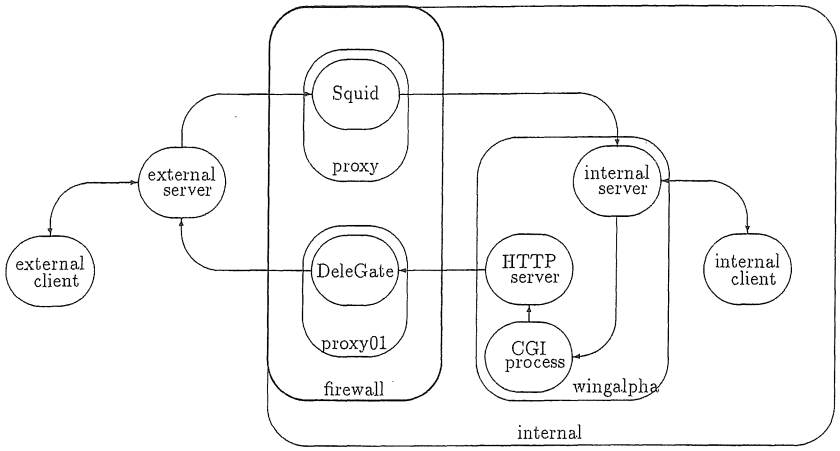


図 3: 代理サーバの通信路

ライアントが待機中のポートへ接続すると他方のサーバがそれに対応するサーバへ接続する．接続が確立したら，双方サーバは受信したデータをそのまま他方のサーバから送信しファイアウォールを超えて接続出来ているかのように見せかける．

4.2 内部サーバの起動

このサービスの管理者が内部サーバ起動用の URI を指定して外部サーバを起動すると，外部サーバは DeleGate を通じて HTTP サーバに接続し，内部サーバ起動用の CGI プログラムの URI を指定する．この時にオブジェクトボディを利用して外部サーバへ接続する為の URI を送り，内部サーバはその応答時にオブジェクトボディを利用して内部サーバへの接続用の URI を送る．これによって，各サーバは各々接続待ち状態となる．

内部サーバを起動する時の要求及び応答の詳細を表 1に示す．プロクシを経由した通信は基本的に HTTP に従う必要がある為，表中では HTTP の要求及び応答時に必要な項目だけ記した．

4.3 サーバ間の通信

サーバ間の通信は，外部サーバから内部サーバは Squid を通し，内部サーバから外部サーバの送信には HTTP サーバから起動された CGI プロセス，HTTP サーバ，DeleGate を経由して外部サーバに送信される (図 3参照)．

内部サーバを起動したら各サーバは他方のサーバの URI へ接続するのだが，接続は切れる場合が頻繁にある．この為，送信したデータは，それが送信された事を確認するまで保存する必要がある．そして，接続が切れた事が検出された時点で再度接続し，送信の終ってないデータを再度送信する．

Request	methed	POST
	URI	外部サーバ起動時に指定
	protocol	HTTP/1.0
	Content-type	text/plain
	Object-Body	外部サーバへの接続用 URI
Response	protocol	HTTP/1.0
	Content-type	text/plain
	Object-Body	内部サーバへの接続用 URI

表 1: 内部サーバ起動 HTTP

Request	methed	POST
	URI	内部サーバ起動時に取得
	protocol	HTTP/1.0
	Content-type	text/plain
	Object-Body	送信要求メッセージ ID
Response	protocol	HTTP/1.0
	Content-type	application/octet-stream
	Object-Body	サーバ間の通信内容

表 2: サーバ間接続要求 HTTP

サーバ間の通信路を確立する為の要求及び応答メッセージを表 2に示す．このやりとりによって，返答する側から要求を出した側への通信路が確立し，応答のオブジェクトボディを利用してデータを送る．

4.4 サーバ間通信プロトコル

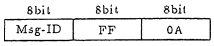
サーバ間の通信プロトコルを表 3に示す．これは，表 2のレスポンス (Response) のオブジェクトボディ (Object-Body) に相当するものであり，接続したのち制御 ID(図中では Ctrl-ID と表記．00 ～ FF の値を持つ) を送り，その後このプロトコルにて通信する．また，先の制御 ID を含む各メッセージの終端には 0A が

付加する．これは，プロクシが 0A を読み込んでから送信する仕様になっている為である．

各メッセージには先頭にメッセージID(表中で Msg-ID と表記．プロトコル上では 00 ～ 7F 可能だが設定による) が付加しており，そのメッセージを受け取ると 80 加算したものを返しそのメッセージを受け取ったことを報告する．これによって，再接続した時にどのメッセージから再送信するか指定する事が出来，信頼性のある通信が可能となる (データそのものの信頼性は TCP による)．

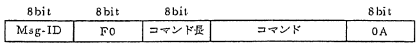
4.4.1 終了メッセージ

終了時のメッセージを以下に示す．このメッセージを受け取った場合に限り応答は一切なく，接続を切断し，プロセスを終了する．



4.4.2 制御コマンドメッセージ

管理者によって外部サーバに制御コマンドが入力された時，以下に示す制御コマンドメッセージを送信してそれを内部サーバにも伝える．



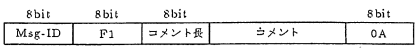
制御コマンドとしてはサービスの設定があり，以下に示す形で与えられる．

```
("enter"|"leave") "␣" <listen-port> "␣"
<connect-address> "␣" <connect-port>
```

“enter” は外部から内部へ接続するサービス，“leave” は内部から外部へ接続するサービスを起動する．listen-port で指示されたポートで接続を待機し，connect-address で示されたアドレスの connect-port で示されたポートへ接続するサービスを提供する．

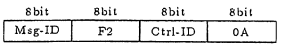
4.4.3 コメントメッセージ

コメント送信時のメッセージを以下に示す．これは内部サーバの状況を外部サーバの管理者に伝えるためのもので，外部サーバはこのメッセージを受け取ると，そのコメントを出力するだけで，特に動作を必要は無い．



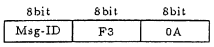
4.4.4 再接続要求メッセージ

プロクシから受信側への通信路が切断されたにも関わらず，送信側からプロクシへの通信路が接続状態のままである場合もあり，再接続する場合にはそれを相手側のサーバに伝え再度接続を行う為に，以下に示すメッセージを送る．



4.4.5 送信テストメッセージ

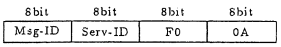
各サーバは送信すべきデータが無い場合一定時間毎に，以下に示すメッセージを送り，通信路のテストを行う．



このメッセージを送って一定時間反応 (メッセージ ID+80 によるメッセージ受信報告) が無い場合，受信が正常に行われてないとみなし，通信路を切断し再接続を試みる．

4.4.6 サービス始動応答メッセージ

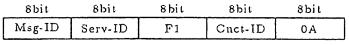
制御コマンドによってサービスが無事始動成功した場合以下に示すメッセージを送り，それを報告する．



サービスの状態変化はサービス ID(図中では Serv-ID と表記．80 ～ BF の値を持つ) を使い，どのサービスか判別する為の識別子とする．サービス ID はサービスが始動する時点で最も小さい空き ID を使う．

4.4.7 接続要求メッセージ

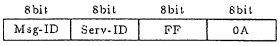
待機中のポートに接続されると，それを知らせ，相手側のサーバでそれに対応したホストの接続する要求をする為に，以下に示すメッセージを送る．



この時のその接続の接続ID(図中では Cnct-ID と表記．0 ～ 7F の値を持つ) を送り，その接続への識別子とする訳だが，メッセージの転送にタイムラグがある為，双方が同じ接続を同じ ID で持つとは限らない．その為識別子としての接続 ID は相手側の接続 (図中では Xcnct-ID) によって指定される．

4.4.8 サービス終了メッセージ

サービスを終了させる時，及び，待機用ポートが空いてなかった場合等サービスの始動に失敗した時，以下に示すメッセージを送る．



Msg-ID	FF	-		終了
	F0	コマンド長	-	制御コマンド
	F1	コメント長	-	コメント
	F2	Ctrl-ID	-	再接続要求
	F3	-		送信テスト
	Serv-ID	F0	-	サービス始動応答
		F1	Cnct-ID	接続要求
		FF		サービス終了
	Xcnct-ID	F0	Cnct-ID	接続応答
		FF	-	接続切断
		データ長		データ送信
Msg-ID+80	-			メッセージ受信報告

表 3: サーバ間プロトコル

4.4.9 接続応答メッセージ

接続要求メッセージへの返答として，接続が成功した場合，以下に示すメッセージを送りそれを報告する．この時相手側の接続 ID を識別子としながら，自分側の接続 ID を相手側が識別子として指定できるようする為必要となる．

8bit	8bit	8bit	8bit	8bit
Msg-ID	Xcnct-ID	F0	Cnct-ID	0A

4.4.10 接続切断メッセージ

接続が切断された場合，及び，接続に失敗した場合以下に示すメッセージを送信してそれを報告する．

8bit	8bit	8bit	8bit
Msg-ID	Xcnct-ID	FF	0A

4.4.11 データ送信メッセージ

接続がデータを受け取ると，以下に示すメッセージによってそのデータを送信する．

8bit	8bit	16bit		8bit
Msg-ID	Xcnct-ID	データ長	データ	0A

5 遅延時間の計測

今回作成した代理サーバは多くのサーバを経由する為に時間の遅延が発生する．ここでは，その遅延時間を計測し，その特性及び程度を調べた．

5.1 実験方法

本来，この代理サーバはファイアウォールの内側と外側のホストでサーバを起動することを目的として設計されているが，ここでは実験の為に外部用サーバをファイアウォールの内部で起動する．これは，外部サーバへの通信と内部サーバへの通信を同じマシンのプロセスが制御することによって正確な時間を測定するためである．

外部サーバを内部で起動し，測定プログラムが外部と内部の両方へ接続する．そして，片方への接続へ送信してそれが他方のサーバより返されるまでの時間を計測する．これを，外部から内部及び，内部から外部ともに 10sec 毎に行い，その統計をとる．

5.2 実験結果

遅延時間の分布を図4，遅延時間の推移を図5に示す．

図3を見て判る通り内部から外部への送信には多くのステップを要する為，より大きなラグが発生している．内部から外部への送信に比べ，外部から内部への送信は平均ラグ時間は少ないが，ラグ時間が安定していない．これは，外部から内部へのアクセスに比べ内部から外部のアクセスの方が遥かに多い為，HTTP プロクシの使用状況に左右され易いことが原因であると考えられる．図5を見て分かる通り，平日の昼間等プロクシの使用頻度の高くなりがちな時間帯により大きなラグが発生している．平均的な遅延時間は往復で 2.3sec 程度となる．これは，Telnet や Rlogin といったインターラクティブなログインサービスに使うにはあまりにも絶望的な値と言える．その反面，ファイルの転送やニュース等といった迅速なレスポンスを必要としないサービスには大きな問題ではないと思われる．

6 まとめ

本論文では HTTP プロクシを利用した通信法を提案し，その実用として代理サーバを作りその性能について検討した．これによってファイアウォールによる接続不能問題をある程度解決の糸口を見つけることは出来たと思われるが，まだこれから解決しなければならない問題は多々ある．

6.1 代理サーバの問題点

今回作成した代理サーバは，ファイアウォールの内部と外部のサーバを起動する訳だが，では外部の何処に起動すればよいのかという問題がある．つまり，外

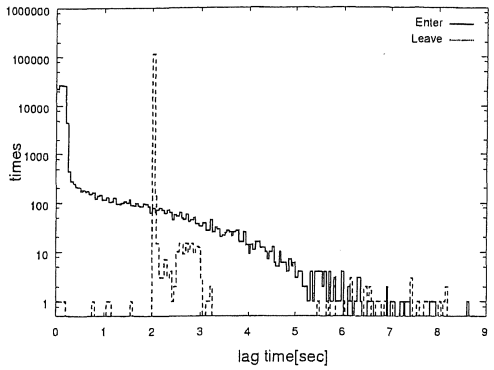


図 4: 遅延時間の分布

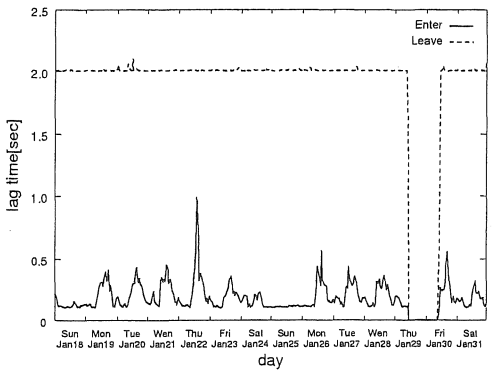


図 5: 遅延時間の推移

部のネットワーク上にサーバを起動できる権限が必要となる.

また, このサーバでは中継するデータの中身を一切見ていないし, 各サービスの接続先は固定される. 前者はそのアドレスやポート番号を送ってそこへの接続を要求するようなサービスへの適合不能性を表し, 後者は WWW の様なアクセスするサーバが多数あるサービスを受ける場合に問題となる. 実際プロクシを作る場合, どの様なサービスに使うか考えて作られるもので, 様々なサービスに対応する為にはそれぞれのプロトコルに対応したルーチンが必要となる.

6.2 今後の課題

今回作成した代理サーバは実験の課程で作られたもので, そのプロトコルを見て分かる通り無駄が多い. 各メッセージの有効性及び無効性, あるいは他にどんなメッセージが効果的か, 深く考えた上でプロトコルを練り直せば, 速度の向上も考えられるであろう.

6.3 IP 中継

今回 TCP で送信されたデータを中継したが, 同様な方式で IP パケットを中継する事も考える事は出来る. この場合の利点として, IP パケットの転送は絶対的な信頼性を必要としない事 (実際のインターネットでも 50%を越えるパケットロスがある場合もある), 外部の全てのサービスを利用できる事, が考えられる. 信頼性を必要としない為データを転送できたか確認する必要が無く, データをバッファに貯める必要もなくなるので, プロトコルの簡略化が出来, 更に, 使用メモリ及び CPU 負荷ともに軽くなる. また, IP パケットが直接送受信出来る為, この場合そのホストがファイアウォールの外にあるのと同等な動作が可能になる.

しかし, IP パケットの送受信を許可されてない環境も少なくなく, TCP によるやりとりの分をこのサーバが行なう為, 遅延時間が更に増える上に, ルータと成る訳だからかなりの回線容量を必要とする. また, 外部のネットワークと直接触れるのと同じ状態になる為, 中継サービスと同時にファイアウォールとしての動作も必要になる. この場合, 中継サーバでありながら, ルータであり, ファイアウォールを兼ね揃えなければ成らない.

7 謝辞

本研究を遂行するにあたって, 本システムが実行可能かつ必要となるネットワークを設計, 管理して頂いた羽賀研究室及び計算センターの方々, に深く感謝申し上げます

参考文献

[1] W.Richard Stevens. 井上 尚司訳. 橘 康雄 訳. 詳解 TCP/IP. ソフトバンク株式会社, 1997

[2] Stephen A.Rago. 小暮 博道 監訳. 詳解 UNIX ネットワーク プログラミング. ソフトバンク株式会社, 1995

[3] William R.Cheswick, Steven M. Bellovin. 川副 博 監訳. 田和 勝, 鎌形 久美子 訳, ファイアウォール. ソフトバンク株式会社, 1995

[4] T.Berners-Lee, R.T.Fielding, Frystyk Nielsen. 木内 貴弘 訳. ハイパーテキスト トランスファープロトコル第 1.0 版仕様書 (日本語版), 1995

(受理 平成10年 3 月20日)