〔研究〕

多目的プロトコル中継システムDeleGate

Multipurpose Protocol Mediation System: DeleGate

佐藤豊
Y. SATO

This paper presents design and implementation of a multipurpose protocol mediation system named DeleGate. DeleGate makes computer network comfortable for users, providing features such as relaying communication through firewall securely for systems and efficiently for users, cache for data and connection that gives quicker response for users and reduces load for network, conversion among protocols which gives many protocols available for a client software of a specific protocol, routing based on pattern matching of domain names which is easily and compactly described by users, and character code conversion to make transfered text data be acceptable by a client software. This system has been distributed on internet and is used at over 600 organization in the world.

§1　はじめに

1994年の1月には，インターネットに接続されたコンピュータの台数は220万台であったが，その半年後には320万台に達した。このような急成長は，インターネットが従来の研究機関による研究利用目的のネットワークから，行政機関，企業，一般家庭までを含んだ商業的利用を許容する実用のネットワークへと変貌する過程で起きている[4]。これに伴い，接続するコンピュータも従来のUNIXワークステーション中心からパーソナルコンピュータが主役になりつつある。また，アプリケーションとしては，WWWやGopherなどの広域的マルチメディア情報探索システムが急速に延びており，これが昨今のインターネットの裾野への広がりを促進している。

予想を越えた急速な大規模化，利用の多様化の進むインターネットは一方で，技術的な面で，またネットワーク接続の整備の面でも，多くの問題に直面することになった。特に，セキュリティ保護と利用者の利便性の両立の問題，不十分な通信容量の問題，サーバ/クライアント間の文字コードの不一致の問題は緊急な解決を必要としていた。

これらの問題の解決に迫られて，DeleGateの開発は始められた。DeleGateはインターネットプロトコル(IP)上の各種アプリケーションプロトコルを，アプリケーション層で中継するソフトウェアである。DeleGateをネットワークの防火壁(fiewall)の上で動作させ，防火壁によって往来が遮断されているIPプロトコルを中継するとともに，一旦中継したデータの保存・再利用による高速化，クライアントの望む文字コードへの変換を行うことで，上述の問題をまとめて解決ないし緩和することができる。

DeleGateの開発は94年3月に始められ，同時に電子技術総合研究所内で試用に供された。その後工業技術院情報計算センターを中心とするAISTネットワーク内での公開を経て，同年6月には所外の組織に対して試用評価のために，インターネット経由(anonymous FTP)での配布を開始，その後幾つかのフリーソフトウェア集CD-ROMにも収められ再配布されている。95年5月現

キーワード: インターネット, プロトコル, firewall, セキュリティ, proxy, キャッシュ, 経路制御，WWW

在，DeleGateは国内外の600を越える組織で利用されており，利用者から得られた評価や改善要求を反映しつつ改良・拡張が続けられている[2)]。

緊急の問題を94年末の第1版最終版においてほぼ解決した後[1)]，DeleGate第2版以降の開発の力点は，中継の性能の向上技術や，アプリケーション層での中継システムによって開ける新たな可能性の模索（それによりどのような付加価値的機能を提供し得るか）に移った。中継可能なプロトコルの広がり，中継の目的の多様化を伴いつつ，DeleGateはより多目的のプロトコル中継システムとして成長を続けている。

本報告では，上述の問題点の解決や可能性の模索のために，DeleGateの現在までの版で実現した機能・方式の概要と，今後の展望を述べる。

§2 開発の背景・動機・目標

DeleGateの開発が始められた1994年は，日本においてGopher[10)]やWWW[†1]の利用が本格化，爆発的に普及し，これによって一般社会に「インターネット」が初めて知られるようになった一年であった。このような爆発の兆しの見え始める94年初頭，電総研においてもまずGopherのサービスおよび利用が開始されたが，それと同時に以下のような問題点に直面することとなった。

- セキュリティの問題
  企業や行政機関まで利用範囲が広まった現在，セキュリティ保護はますます重要になっている[4)]。そこで保護のために組織内部と外部の直接的な通信を遮断する「防火壁(firewall)」を設けるのが一般化しているが，これにより防火壁の外(内)にあるサーバへのアクセスが不便になる。

- 通信容量の問題
  WWWやGopherにより，画像や音声を含むデータ量の大きなマルチメディア情報が対話的に利用されるようになった。しかし多くの組織がインターネットへ接続する通信回線の容量は，これらのデータの転送には不十分であり，特に多数の利用者からのアクセスが集中した場合には実用に耐えないものとなる。

- 文字コードの問題
  日本語テキストを表現するための文字コードでは，パーソナルコンピュータの標準的文字コードとインターネットの標準コードは異なる。このため，インターネット上のサーバの提供する情報の文字コードと，パーソナルコンピュータ上のクライアントの解釈可能な文字コードが一致せず，表示ができない。

DeleGateではこれらの問題点を解決するための，アプリケーション層における中継ソフトウェアとして開発された[†2]。IP層の専用ルータではなく，アプリケーション層のソフトウェアで実現することにより，ルータ等の特別な設備投資は不要であり，上記の問題をまとめて解決することができる。

DeleGateの短期的な目標は上述の3つの問題をまず解決することであったが，アプリケーション層でのソフトウェアによる中継システムとしての特長を活かして，以下のような高次機能への展開が可能であると考えた。

- 賢い経路選択
  アプリケーションプロトコル毎の内容に応じたきめの細かい経路制御や，ドメイン名を用いた簡潔な制御記述。

- 非**IP**経路による中継
  IPでない通信経路，たとえば低速な電話回線や高速なATMなどを有効に利用できるプロトコルの上での，各種アプリケーションプロトコルの中継。

- プロトコル開発のためのプラットフォーム
  複数のプロトコルを統合的に中継するソフトウェアの延長として，既存のプロトコルの整理・統合・拡張を支援し，今後の新たなアプリケーションプロトコル開発のための共通のプラットフォームを提供する。

[†1] スイスのCERNで始められたハイパーテキスト型広域情報探索システムのプロジェクトWorld Wide Webの頭文字をとったもの。WWWではサーバ/クライアント間の通信プロトコルとしてHTTP (Hypertext Transfer Protocol)[6)]を開発，HTTPの上で交換されるメッセージにはインターネットメイルと同じMIME(Multipurpose Internet Mail Extensions)形式[11)]を採用している。ハイパーテキストの記述にはSGMLをベースとするHTML(Hypertext Markup Language)[9)]が，HTML中で他の情報を指し示すのに使われる識別名としてURL(Uniform Resource Locators)[8)]が開発された。クライアントプログラムとしてはイリノイ大学NCSAで開発されたMosaicがWWWの隆盛をもたらし，現在はNetscape社のNetscape Navigatorが主流となっている。

[†2] 開発開始当初はGopherプロトコル中継専用のサーバとして*go-far*と名付けられ，その後すぐに汎用の中継サーバに拡張されてDeleGateと改名された。

開発を進める中で得た知見をもとに，また多数の利用者からの活発なフィードバックを得て，当初に設定したこれらの目標を越えた機能が実現され，また 4章に述べるような新たな展望が開けつつある。

§ 3 DeleGate の概要

現在までに実現された DeleGate の主要機能を図 1に，中継可能なプロトコルおよびデータ形式を表 1に示す。

- 経路制御 (全プロトコル共通)
  アクセス元 / アクセス先のホストのドメイン名または IP アドレスのパターンマッチングにより，中継先を選択，あるいは不通の経路を迂回する機能。
- アクセス制御 (全プロトコル共通)
  アクセス元 / アクセス先のホストおよびユーザ名により，アクセス許可 / 不許可を制御する機能。ユーザ名は Ident プロトコル[15)]を用いて取得する。
- プロトコル変換 (Gopher,WAIS,FTP → HTTP)
  各種のプロトコルで提供されている情報を，HTTP プロトコルに変換してクライアントに提供する機能。
- 文字コード変換 (Telnet,X,CU-SeeMe 以外)
  中継データに含まれるテキストの文字コードを，サーバ / クライアントで受理可能なものに変換する機能。
- **MIME** コード変換 (NNTP,SMTP,POP)
  MIME 形式による，メイル / ニュースのヘッダ部・ボディ部の非 ASCII テキスト，バイナリーデータのエンコード / デコード機能。
- データキャッシュ (HTTP,Gopher)
  サーバからクライアントへ一度中継したデータを保持し，同一データへのアクセスの際に再利用する機能。
- コネクションキャッシュ (FTP/HTTP)
  一旦接続したサーバへの接続を維持し，複数のクライアント間でその接続を共有・再利用する機能。
- 既存ソフト用中継 (HTTP,Gopher,FTP,Telnet,POP)
  既存のクライアントソフトウエアに対して変更を加えることなく中継機能を提供する機能。
- マウント機能 (HTTP)
  サーバ / クライアント間の双方向の通信に含まれる情報の識別子 (URL) を書き換え，実際の情報の所在位置を秘匿してクライアントに提供する機能。

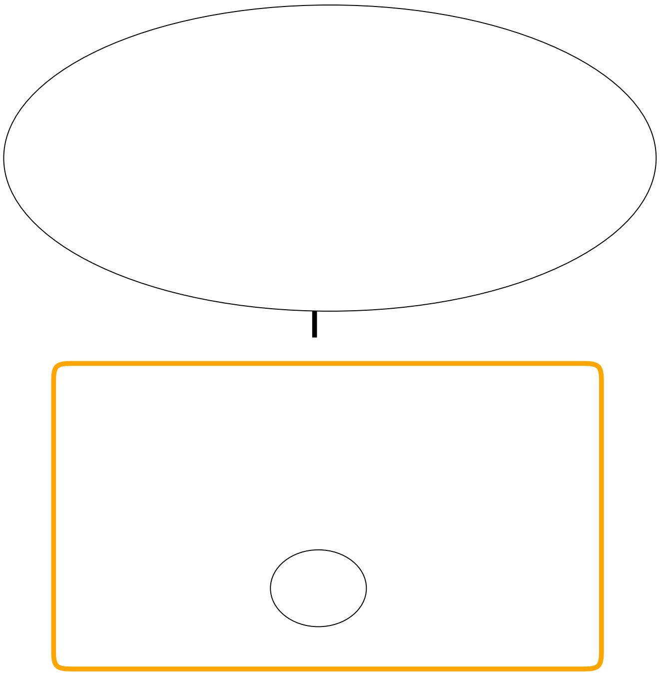


図 1 DeleGate の主要機能

表 1 DeleGate が中継するプロトコルとデータ形式

名称	機能概要	規格名
HTTP	WWW の通信プロトコル	Internet-Draft
Gopher	広域分散情報探索	RFC1436
WAIS	広域分散情報検索	RFC1625
Whois	人名ディレクトリサービス	RFC954
FTP	遠隔ファイル転送	RFC959
Telnet	遠隔ログイン	RFC854
X	遠隔ウィンドウシステム	RFC1013
NNTP	ニュース配送	RFC977
SMTP	メール配送	RFC821
POP	遠隔メールボックス管理	RFC1460
CU-SeeMe	簡易遠隔会議システム	
MIME	多言語多目的メール形式	RFC1521
HTML	ハイパーテキスト記述言語	Internet-Draft
URL	汎用広域情報識別名形式	RFC1738

この他に，Ident(利用者認証プロトコル，RFC1413) 使用

本章では以下，これらの主要機能について概説する。

3.1 防火壁を越えて – 内から外へ –

DeleGateの最も基礎となる機能は，アプリケーション層においてアプリケーションプロトコルを中継する機能である。これはセキュリティ保護を実現するための機能であるとともに，中継に伴ってDeleGateが提供する各種の付加価値的な機能を実現するための前提となる機能である。

3.1.1 セキュリティ保護と利便性の問題

インターネットに接続することによって世界中のコンピュータと交信が可能になる。これは同時に，自らのコンピュータが世界中からアクセス可能になり，悪意ある攻撃の危険に晒される事を意味する。しかし，これに備えて組織内の全てのコンピュータのセキュリテイレベルを厳重に管理することは，人的コストが高く困難である。そこで，組織外部と直接通信が可能なホスト(あるいはネットワーク)を限定し，組織内(外)のコンピュータは必ず一旦そのホストを経由しなければ外部(内部)と交信できないようにする方法が取られるようになった。このようなホストは一般に防火壁(*firewall*)と呼ばれる。防火壁を境として，外部と内部の間のIPパケットの中継経路を遮断することで，悪意の侵入や，事故による漏洩などの危険を低減することができる。

防火壁の構築により安全性は向上するが，引き替えに利用者の利便性が損なわれてしまうという問題が生ずる。例えば，エンドユーザが外部にあるFTPやWWWなど各種のサーバにアクセスしようとする時には必ず一旦Telnetなどによりfirewallにリモートログインし，そこで応用プログラムを実行しなくてはならない[†3]。これは不便であるばかりでなく，利用可能なサービスやクライアントホストの機種が制限されてしまう[†4]。

3.1.2 従来のプロトコル中継システムによる解決

この問題を解決するために，ソフトウェアで実現した中継システムにより特定のアプリケーションのプロトコルだけを，限定された利用者やアクセス対象についてのみ中継する方法が開発されて来た。それらは，特定のプロトコルに依存しない汎用のものと，個別のプロトコルを対象としたものに分けられる。

汎用中継システム

代表的なシステムとして，TCP/IP上のプロトコル全般を中継する*SOCKS*が広く使われている[14)]。これは，防火壁の上で汎用の中継サーバ(*sockd*)を実行し，各クライアントプログラムに変更を加えてこのサーバに中継を任せるようにする方法である。SOCKSを利用するには，クライアントプログラム中にあってIP通信の出入口となる*socket*関数群をSOCKS用の*reliable*なsocket関数群で置き変えればよい。Unix向けのSOCKS関数ライブラリが公開されており，telnet, ftp, Mosaicなど多くのクライアントプログラムに組み込まれている一方，パーソナルコンピュータの方面での普及は遅れている。

一方，UDP/IP上のプロトコル全般を中継するものとしては，*udprelay*がある[19)]。

個別プロトコル中継システム

代表的なシステムとしてHTTPプロトコル中継機能付きの*CERN-httpd*，いわゆるProxy(代理サーバ)[7)]が広く使われている。クライアントとProxyとの交信プロトコルはHTTPであるが，HTTPではクライアントからの要求の対象を示すのに用いられるURLの中に対象へのアクセス手段となるプロトコルを表すことができ，またサーバからの応答データを表現するMIME形式で各種のデータ型を表すことができる。これによってProxyはHTTP以外にもGopher, FTP, WAISなどを中継することができる。この場合Proxyは，Gopher, FTP, WAISなどのサーバと交信して得た応答を，MIME形式に変換してクライアントに返す。このProxy方式に対応するために，クライアントには(1)要求を目的のサーバでなくProxyサーバに送る，(2)要求として目的のサーバとポートの情報を含むURLを送る，という変更を加えてやれば良い。

3.1.3 従来方式での不足と**DeleGate**による解決

SOCKSの方式は汎用であるが，各プロトコルの内容を解釈して中継するものではなく，2章に述べた各種の問題のうち，セキュリティの問題しか解決しない。一方，ProxyによるHTTPプロトコル上での「URLによる要求とMIMEによる応答」の方式は，SOCKSのような汎用の中継システムに比べて適用できるプロトコルの範囲が狭い。すなわち，HTTPプロトコルのクライアントだけを対象とし，「要求 / 応答型」でないプロトコル，例えばTelnetプロトコルなどには適用できない。

[†3] 電総研ではこのような「出島方式」により，WIDEインターネットに接続している。

[†4] 例えばfirewallがUnixマシンであり，内部のMacintosh上のMosaicから外部にアクセスしたい場合など。

2章に述べた問題は，Proxy 方式でカバーできない各種のプロトコルにも共通して存在するものであり，これらを包括的に解決する方式が望まれる。DeleGate はそのような方式を実現するものである。

DeleGate は，SOCKS のような汎用の中継システムとしても，Proxy のような専用の中継システムとしても利用することができる[†5]。中継システムとしての DeleGate は，以下のような特徴を持つ。

- CERN の Proxy と互換の中継機能を提供する (Proxy の代わりとして利用可能)
- Proxy 方式の対象外のプロトコル，あるいは Proxy 非対応のクライアントも中継可能
- 単一のサーバ専用の中継サーバとしての利用が可能 (X プロトコル等)
- サポートする全てのプロトコルに対する共通の方法でのアクセス制御，経路制御機能を提供

DeleGate は Proxy 方式に非対応の HTTP クライアントに対しても中継機能を提供する[†6]。これは DeleGate がクライアント / サーバ間の双方向の通信内容をともに解釈して必要な書き換えを行うことで実現されており，これにより後述 (3.5) の「マウント機能」も実現されている。

3.1.4 ルータによる解決法との比較

従来，ネットワークのセキュリティ保護は，IP 層のルータにより実現されて来た。

- 専用ルータによる実現
  IP パケットの経路制御を行うルータの多くは，パケットの送信元，送信先，プロトコルの組合せにより，指定したパケットを通過させない機能 (フィルタ機能) を持っている。
- **IP パケット濾過サーバによる実現**
  専用のルータではなく Unix システムをルータとして利用している場合，Unix の基本システム (カーネル) に変更を加えて，IP パケットをフィルタする *screend* などのソフトウェアが用いられている[13)]。

これらの方法では，高速な中継が可能である反面，設置にかかるコストは高く，柔軟な変更や設定にも向かない。これに対して前述のようなプロトコル中継システムでは，その逆の利点と欠点を持つ[†7]。ただし，この両者についてはどちらをとるかという問題ではなく，むしろいかに組み合わせて必要な性能と機能をトレードオフするかという問題である。

ルータにフィルタ機能が無く，screend なども用いない場合には，外部と内部との連絡を全て自由に通すか，全て遮断するかの選択となる。全て遮断する場合には，firewall マシンだけが外部と内部の両者に直接交信できるホストになるので，ここで中継サーバを実行する。アプリケーション層の中継システムでは，もともと内部と外部とで IP パケットの直接通過が不可能な状況，例えば内部でプライベート IP アドレス[12)]を用いている場合にも中継ができる。中継をアプリケーション層で行うので IP 層での透過性を必要としないし，そもそもサーバ / クライアントの通信プロトコルが IP である必要性も無い。

いずれにしても，アプリケーション層のソフトウェアで中継を行う技術は，以降に述べるような様々の，中継に伴う付加価値的機能を実現するための基盤として欠かせない技術である。

3.2 データキャッシュによる高速化

組織としてインターネットに接続するための回線速度としては，日本国内では今でも 64Kbps の専用回線あるいは ISDN 回線が一般的である[†8]。このような低速な回線でも，実時間性への要求が低い電子メイルなどのアプリケーションでは十分実用に耐えた。しかし，対話的な双方向通信を必要とする WWW などを，多人数が同時に利用する状況にはとても耐えることができない。

[†5] DeleGate は，SOCKS と組合せて (SOCKS のクライアントとして) 動作させたり，CERN の Proxy と組合せて (Proxy のクライアントとして) 使用することもできる。従って，これらの中継サーバが既に firewall に装備されているなら，改めて firewall 上に DeleGate をインストールしなくても，DeleGate を利用することができる。

[†6] DeleGate の開発ではまず HTTP や Proxy 機能を前提としない中継機能が作られ，Proxy と互換の中継方式は後に追加された。

[†7] ネットワークが 10Mbps 程度なら，アプリケーション層の中継がスループット上のネックになることはないが，中継サーバの介在によってレスポンスが数ミリ秒～十数ミリ秒程度悪化するのは避けられない。

[†8] 電総研においても昨年までは，数百人からの利用者が WIDE インターネットへの 64Kbps の専用回線で賄われていた (現在では 192Kbps に増強されている)。さらに工業技術院では，傘下の 15 研究所の通信全てが，TISN への 64Kbps 専用回線で賄われていた (現在では 6Mbps で IMnet に接続している)。

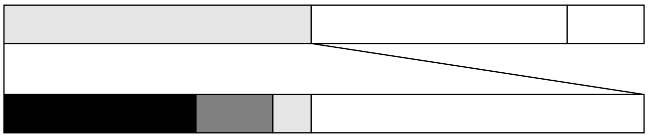


図 2 キャッシュの効果

HTTP を例に取ると，平均的な数キロバイトのテキストページには足りるが，少し大きな画像や音声データでは実用的な応答性が確保できない。例えば，葉書サイズ程度のカラー画像は一般に 100K バイト (800K ビット) 程度となり，これを送るのに 64Kbps の回線では，回線を占有できたとしても 12 秒を要することになる。これでは，対話的な利用とはとても言えない。

そこで，低速な回線をカバーするための自然な方法として，キャッシュ (*cache*) が導入された[†9]。これは，一度得たデータをローカルなキャッシュに保存しておいて，同じデータに二度目にアクセスした場合にはそれを再利用するものである[†10]。特に，多数のユーザが同一データに繰り返しアクセスする場合に，遠隔のサーバへ毎回アクセスする無駄を省くことができる。これにより，エンドユーザに対して高速な応答を与えるだけでなく，転送経路にあるネットワークへの負荷を低減させる効果もある。キャッシュされたデータへは，LAN の速度でアクセスできる[†11]。

表 2 HTTP 用 DeleGate の利用状況とキャッシュの効果
電総研情報系共用計算機 etlcom 上の DeleGate において
(A) 1994 年 10 月 31 日 (月) ~ 11 月 2 日 (水) の間測定
(B) 1995 年 4 月 18 日 (月) ~ 4 月 20 日 (水) の間測定
ともにキャッシュ保存期限 30 日間，キャッシュ有効期限 24 時間

	(A)		(B)	
a. クライアントホスト数	145		198	
b. (ホスト名不明分)	(26)		(81)	
c. サーバホスト数	583		1365	
d. 総アクセス回数	25339		34278	
e. ローカルサーバ参照	10286	40.6%	5350	15.6%
f. 内部データ参照	552	2.2%	1713	5.0%
g. エラー	386	1.5%	543	1.6%
h. キャッシュ対象外	347	1.4%	671	2.0%
i. キャッシュ無視	1673	6.6%	237	0.7%
j. キャッシュ参照	12095	47.7%	25764	75.2%
キャッシュ参照の内訳				
k. キャッシュミス	6306	52.1%	12879	50.0%
l. キャッシュヒット	5789	47.9%	12885	50.0%
m. 有効期限内	3725	30.8%	7636	29.6%
n. 有効期限切れ後変更無	1396	11.5%	4295	16.7%
o. 有効期限切れ後変更有	668	5.5%	954	3.7%
p. 有効キャッシュヒット	5121	42.3%	11931	46.3%
q. 総データ転送量 (Mbytes)	96.5		386.3	
r. 実データ転送量 (Mbytes)	65.7	68.1%	250.9	64.9%
s. リモートサーバ参照	10776	42.5%	19579	57.1%

[j=k+l, l=m+n+o, p=m+n, s=g+h+i+k+n+o]
b. 動的に割り当てられた IP アドレスを含む
e. LAN 内にあるローカルサーバについてはキャッシュを行わない
f. DeleGate サーバ内に組み込まれているアイコンデータなど
g. 接続不可，アクセス拒否など
h. PUT,POST などの非キャッシュ対象の HTTP メソッド，FTP アクセスなど
i. クライアント側からの指令によりキャッシュを利用しなかった場合
n. 有効期限切れの場合は実際のサーバの元データを探索するが，
その結果 DeleGate のキャッシュに保存した日付以降元データに
変更が無いとわかった場合には，キャッシュのデータを再利用する

ここでキャッシュの内容は，元のデータが変更されたなら，無効にされなければならない。このような場合のために，HTTP や Gopher+ のプロトコルでは，データの最終更新日付を調べる機能が組み込まれている[†12]。しかしながら，低速な回線と多段階のルータの経由により，遠隔のサーバとの接続はそれ自体が極めて低速であることが多い。例えば数秒 ~ 十数秒程度を要する事も少なくない。ま

[†9] キャッシュ機能を持つ中継サーバは，一般に *caching proxy* と呼ばれる

[†10] もともとキャッシュ方式は，コンピュータの CPU と主メモリの間に小型の高速メモリ (cache) を置き，ここに主メモリの内容のコピーを置いて CPU に高速アクセスさせるものである

[†11] 現在の一般的な LAN の速度は 10Mbps（100Mbps への移行期にある）。クライアントが DeleGate サーバと同一マシン上にあれば，ディスクの読み出し速度でアクセスできる。この場合，一般に律速段階は転送速度では無く，クライアントにおける表示処理の速度になる。

[†12] HTTP では要求ヘッダの中に「If-Modified-Since: 日付」を指定すると，指定した日付以降に更新がなされた場合にのみ応答データが返され，更新がされていない場合にはその旨を知らせる状態コードだけが返る。Gopher プロトコルの拡張版である Gopher+ では，infomation を要求すると，応答中の Mod-Date というフィールドで更新の日付を知ることができる。ただし現時点では Gopher+ のこの機能をサポートしているサーバは多くない。

表 **3** キャッシュのためのディスク消費量

電総研情報系共用計算機 etlcom 上の DeleGate において
(A) 1994 年 11 月 3 日測定
(B) 1995 年 5 月 3 日測定
(ともにキャッシュ保存期限は 30 日間)

	(A)		(B)	
	HTTP	Gopher	HTTP	Gopher
サーバホスト数 (*1)	1128	496	3078	237
ディレクトリ数	6489	3326	18674	1273
ファイル数	15840	5095	56331	1666
ファイル使用量 (Mbytes)	229	241	1023	60

(*1) 非標準ポートの使用による重複したカウントを含む。

た，先方のサーバにアクセスが集中していて接続が拒否されることもあり，さらには先方のサーバが稼働していない場合もある。このために，更新日付を調べるだけでもエンドユーザにとって耐えがたい時間がかかってしまったり，アクセスそのものが出来ないという事態が起こる。そこで，データを取得してから一定の期間を「有効期限」として，その期限が来るまでは更新のチェックを行わず，無条件にキャッシュを使用することとしている。

表 2および図 2に，電総研で実際に運用している DeleGate のキャッシュの利用状況を示す。 a. のクライアントホスト数には，個人使用の計算機も，数十人で利用する共用計算機も区別なく含まれている。最も少なく見積もっても利用者数は 120 人以上である。全アクセス数 d. から，キャッシュを利用しない場合 (e. ~ i.) を除いた数が，実際のキャッシュへの参照数 j. である。このうち，キャッシュに求めるデータが存在した (ヒットした) 場合が l. であるが，有効期限切れかつ元データが更新されていた場合 o. はキャッシュは無効である。キャッシュが有効期限内であった場合 (m.)，あるいは有効期限が切れていたが元データは更新されていなかった場合 (n.) の和 p. が，キャッシュが有効に働いた場合である。 (A) と (B) ではキャッシュの参照比率は大きく異なるが，有効なヒット率 (p/j) は 40 ~ 50% の間にある。また，データ転送の量は 30 ~ 40% の削減となっている。

表 3には，このキャッシュのために使用しているディスクの消費量を示した。 1ヵ月分の保存で，およそ 1 ギガバイトである。これは，このキャッシュによって得られる高速化，恩恵を受ける利用者の数，通信量の削減の効果を考えると，十分に安価であると言えよう。

3.3 文字コードの適応

3.3.1 日本語文字コードの混乱

日本語テキストの表現には，歴史的な理由により，3 種類の異なる文字コードが用いられている。日本語の漢字やかなを表す文字集合は，JIS 規格に規定されたものが共通に用いられているものの[†13]，これを ASCII の制御文字や英文字などと混合しつつバイト列として表現するためのエンコーディング方式として，3 種類の方式が用いられているのである。

- 7 ビット JIS(JIS): インターネットでの公式名は ISO-2022-JP[16]。 ISO-2022 のエスケープコードにより，ともに 7 ビットで表現された ASCII および JIS-X-0208 文字セットを切り替える。主に Unix で用いられ，インターネット上のメールやニュースで流通する日本語テキストの標準形式。
- EUC: ASCII を 7 ビットで， JIS-X-0208 を 8 ビットでエンコードしたもの。主に Unix において，プログラム内部での表現，あるいは組織内部で日本語テキストファイルを保存する場合などに用いられる。 FTP などにより，インターネット上で交換されることもある。
- シフト JIS(SJIS): MS 漢字コード。 JIS-X-0201 と， JIS-X-0208 の文字コード値をシフトした 8 ビット表現で，7 ビット表現の ASCII コードと混合したもの。パーソナルコンピュータにおける日本語テキストの標準形式。

このような複数のコードの並立によって，これまでにも様々の場面においてテキストの交換に支障をきたしてきた。そこへ昨今の WWW の普及により，従来疎遠であったパーソナルコンピュータの世界と Unix の世界の間での，IP による直接的なテキスト交換が始まった。当然の結果として，サーバとクライアントにおける文字コードの不一致の問題が顕在化したのである。ほとんどの WWW サーバは JIS コードでサービスをしており，一方，クライアントの大半を占めるパソコンのクライアントは SJIS コードにしか対応していない[†14]。数多くのクライアントを，複数の日本語コードに対応するよう変更し，今後維持

[†13] JIS-X-0201 で 1 文字 1 バイトで表現されるローマ字 / 仮名文字の集合を，JIS-X-0208 で 1 文字 2 バイトで表現される仮名文字や漢字の集合を規定している。

[†14] 現状でほとんどのネットワークソフトウェアは外国産であり，オリジナルの版から日本語コードに対応しているものは稀である。 Unix ワークステーション系では，NTT の高田氏 < *takada@seraph.ntt.jp* > らが X Window 用 Mosaic に対して，各種日本語コードを始め各国の文字コードに対応可能な拡

して行くコストは高い。また，それぞれに歴史的な背景を持つ文字コードを，いずれか一つに統一することは容易ではない。そして，現在の3種類のコードに加えて，それらとはエンコーディング方式だけでなく文字セットも異なるUnicodeあるいはISO-10646が加わろうとしている。

3.3.2　DeleGateによる文字コード変換

このようなサーバとクライアントの日本語コードの不一致を緩衝するために，DeleGateでは，テキスト型の情報の中継時に，その日本語コード変換を行う機能を内蔵している[†15]。任意の日本語コードから，指定されたJIS, SJIS, EUCいずれかへの変換を行うことができる。

さらにHTTPプロトコルのクライアントに対しては，テキストの一文字一文字を画像に変換してHTMLに埋め込むCII (Character by Inline Image) 機能を持っている[†16]。これを用いると，日本語文字の表示機能を一切持たないHTTPクライアントにおいても，日本語テキストが表示できる。日本語文字以外に関しても同様である。

この機能の実現上の問題の一つは，中継している情報がテキスト型の情報であるか否かの判別である。現在は，HTTPあるいはGopherプロトコルにおいてテキスト型と明示されたデータ，およびNNTP, SMTP, POPのようにテキスト型のデータを基本するプロトコルに関してのみ，変換を行っている。

もう一つの問題は，クライアントの受理可能な文字コードをどのように判別するかである。現在のところ，SJISへの変換用，EUCへの変換用などのように専用のDeleGateを起動して，クライアント側でそれらを選択するようになっている。しかし，これでは多くのDeleGateサーバを起動して管理することが必要になってしまう。ここでWWWに関しては，HTTPプロトコルの枠組の中で，あるいはその拡張により，単一のDeleGateを複数の文字コードへの共用の変換サーバとして利用できる可能性がある。すなわち，(1) クライアントからの要求を見てその種別を知り受理可能な文字コードを推定する[†17]。(2) 要求の中に受理可能な文字コードを明示するような拡張をHTTPに導入する。DeleGate第1版では，(2) を想定した実装を試みに行った[†18]。

3.4　既存クライアントのための中継

DeleGateでは「中継サーバ」の概念を持たないプロトコル，あるいは中継機能をサポートしていないクライアントプログラムに対しても，中継機能を提供する。

TCP/IP上のプロトコルでは，交信の開始にあたってクライアントから目的サーバに対して接続を行う必要がある。従って，一つのクライアントから複数の目的サーバへ順次アクセスするには，本来そのクライアントは，その度に接続をし直さなければならない。一方これを中継サーバにより中継する場合，クライアントは中継サーバにのみ継続的に接続し，中継サーバが順次目的サーバへの接続をし直すことになる。しかし，クライアントプログラムが何らかの中継機能をサポートしていない場合には，中継サーバは目的サーバの切り替えに関する情報を得ることができない。そこでDeleGateでは，中継するアプリケーションプロトコル上を流れるコマンドやデータを解釈して，そこに実際の目的サーバに関する情報を強制的に埋め込むことにより中継機能を実現する。

3.4.1　要求/応答型のプロトコルの中継

HTTP，Gopher，WAIS，Whoisのような要求/応答型のプロトコルでは，要求，および応答の中に含まれる情報の識別名を書き換えることにより，クライアントのプログラムを変更することなく中継を実現できる。以下にHTTPでの例を示す。

まず，目的サーバのホスト: ポートをhostT:portTとし，DeleGateサーバ (delegated) のそれをhostD:portDとする。delegatedは，サーバからクライアントへの応答中に含まれる (a) のようなURLを，(b) のように書き換える。

(a) `http://hostT:portT/path`
(b) `http://hostD:portD/-_-http://hostT:portT/path`

張を行ったもの5)が主に使われて来た。パーソナルコンピュータ系では，95年4月に発表されたNetscape1.1Nが，オリジナルの版から複数の日本語コードに対応した（が，β版から公式版になった現在でも幾つかの実用上の問題が残されている）

[†15] 第1版のDeleGateの普及を促進したのは，この日本語コード変換機能であったと思われる。

[†16] ブルガリアのValentin Shopov氏 < *valentin@rs.aubg.bg* > により提供された‘CIILIB’を用いる。

[†17] HTTPでは，クライアントからの要求の中のUser-Agent: フィールドにクライアントの種別を示す。

[†18] HTTPでは，クライアントからの要求の中に「受理可能な言語」を示すAccept-Language: というフィールドがあり，この中に (charcode=SJIS) のような記述を行う。HTTPはRFC822準拠なので，( ) に囲まれた部分はコメントであってサーバには意味を持たないものある。規格に準拠しつつ実験的に拡張を行うために，このようなエンコードを試みた。

このように書き換えられたURLをたぐってクライアントが目的サーバにアクセスする時，クライアントはhostD:portD，すなわちdelegatedを目的サーバだと考えて接続し，

```
GET /-_-http://hostT:portT/path
```

という要求を送る。これを受けてdelegatedは，真の目的サーバ hostT:portT に対して接続し，

```
GET /path
```

という要求を送る。そして，目的サーバからの応答中に含まれるURLを，上述のように(a)の形から(b)の形に書き換えて，クライアントに中継する。

これが繰り返されることにより，クライアントから目的サーバ hostT:portT へのアクセスは必ず hostD:portD の delegated を経由することになる。

この中継方式は，HTTPに関してはCERN-Proxy方式とそれに対応したクライアントの普及に伴いその役割を終えつつある。しかし，Gopher専用クライアントのための中継のためには，依然としてこの方式が必要である[†19]。またGopherに限らず，中継サーバの指定機能がもともと組み込まれていない他のプロトコルに対しても，プロトコルの拡張やクライアント/サーバの作り換えを行うことなく，この方法で中継機能を導入することができる。DeleGateの現在の版では，Gopherの他にもWhoisに対してこの方法を用いた中継機能を実現している。また，WAISに対してもこの方法で中継が可能である。

3.4.2 ログインコマンドの互換拡張による中継

FTPやPOPのように，目的サーバへの「ログイン」を行うプロトコルには，クライアントからサーバに対してログインのためのユーザ名を送信するコマンドがある。ここで，通常は目的サーバ内のローカルな名前として表現されるユーザ名を拡張して，ホスト情報やポート情報を埋め込むことができる。具体的に，FTP, POPにはともに「USER username」という形式のコマンドがある。そこでDeleGateでは「USER username@host:port」の形式のコマンドを受理するようにした。クライアントプログラムの利用者はDeleGateに接続した後，このように拡張されたログイン名を送ることによって実際の目的サーバに接続する。

3.4.3 その他のコマンドの互換拡張による中継

FTPには，目的サーバ上でディレクトリを移動するためのCWD(Change Working Directory)というコマンドがある。CWDコマンドの引数はディレクトリ名であるが，「//」で始まるディレクトリ名は冗長である。このような無意味なディレクトリ名は，通常のクライアント/目的サーバ間の通信では使われないと考えてよい。そこでDeleGateでは，「CWD //host:port」という形式のコマンドを受理して，指定された目的サーバへの切り替えを行う機能を実現した。

3.4.4 明示的な中継サーバ

Telnetでは，ログインの方法は，サーバやクライアントの種別に依存しており，Telnetプロトコル上では規定されていない。従って上述のような方法をとることが困難である。そこで，利用者が陽に意識する中継サーバとして，Telnetの中継サーバを実現した。利用者がTelnet中継用のDeleGateサーバに接続すると，そこでDeleGate固有のモード（目的サーバ名の入力モード）となり，利用者が目的サーバの名前を入力すると接続が行われる。TelnetおよびFTPに対して，この方法を実現した。

3.5 防火壁を越えて – 外から内へ –

3.1では防火壁を越えて，クライアントが内から外へアクセスする手段としての中継システムについて述べた。一方その逆の場面，すなわち防火壁の中のサーバを外部に公開する手段も必要とされている。このための手段として，SOCKSを利用することができるが，そのために内部のホストのアドレスや名前を外部に晒してしまうという点でセキュリティ上の難点がある。また，CERN-Proxyはこの様な利用方法に一般的に対処することができない[†20]。

これに対してDeleGateでは「マウント機能」によりこのような利用を可能にした。これは，別の（複数の）ホス

[†19] Gopherの場合には，*Directory Entity*型の応答に含まれるホスト名とポート番号をdelegatedの物に書き換え，本来のホスト名とポート番号を*selector*の中に埋め込む。

[†20] Proxyは要求中のURLを書き換えて中継する機能を持つので，限定された状況，つまり内部のサーバが一つだけでHTMLデータ中のURLの参照が全て相対パスで書かれているなどの場合には，DeleGateのマウント機能と同様に利用できる。しかし，例えば内部のサーバから内部のサーバへの参照がある場合に対処できない。すなわち，内部のサーバへのポインタがそのまま「輸出」されてしまう。

ト上にある情報が，利用者から見ると（全て）DeleGateサーバのホスト上にあるかのように見えるようにするものである[†21]。

マウント機能を利用して，内部にある(複数の)サーバホストの情報をfirewallホストにマウントする。すると，外部のクライアントからは全ての情報がfirewall上にあるように見えて，アクセスすることができる。実際には，クライアントの要求はDeleGateにより内部のサーバに中継されて処理され，その応答が再びDeleGateを経由してクライアントに返される。DeleGateはこの双方向に中継するプロトコルの内容を知っており，

(1) 要求の中継に際して，要求された情報の名前から内部のサーバを判定して そのサーバへの要求に書き換える

(2) 応答の中継に際して，応答中に含まれる内部サーバへの参照を，firewall上にある情報への参照であるかのように書き換える

という処理を行っている。HTTPサーバのマウントの例を図3に示す。この機能は，3.4.1で述べた双方向の書き換えによる中継機能の上に実現されている。

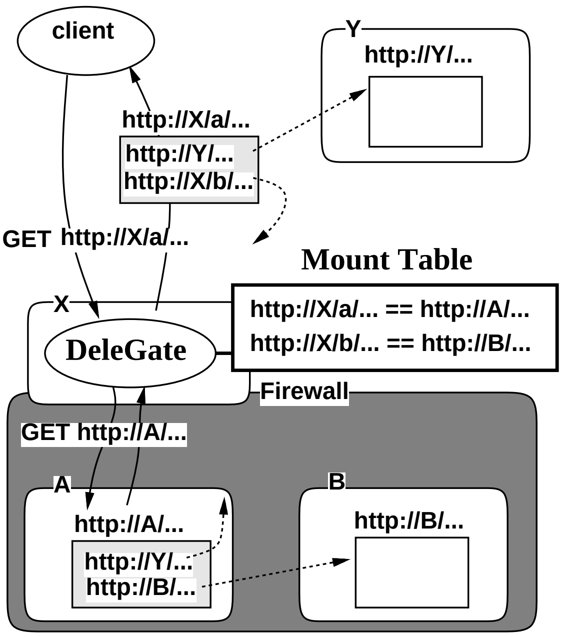
マウント機能は，このような防火壁越えのための利用だけでなく，一般にあるサーバの情報を別のサーバから中継しながら，その中継をクライアントに意識させないようにするのに有効な方法である。例えば，4章に述べるようなFTPのミラーサーバの実現や，複数のNNTPサーバを統合してサービスする場合などにも応用することができる。

3.6 経路制御

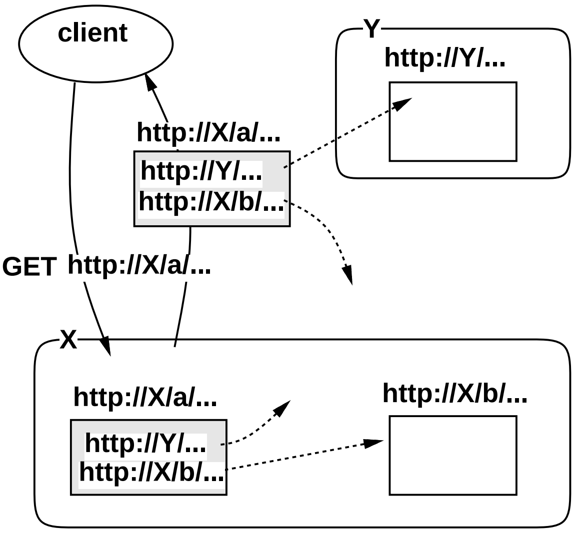
あるホストから目的のホストへ達する通信経路が複数存在する場合は少なくない。このような状況では，最適の経路を選択することによって通信の高速化や，障害経路の迂回，各経路への負荷の分散ができることが望まれる。

多重の経路はまず，広域ネットワーク間の相互接続が，木構造状ではなく網の目状に行われることによって起こる。また，一つの組織が複数の広域ネットワークに接続することによっても生じる。

一般にIP層のルータは，このような複数の経路の中から，目的のホストごとに固定的に決まる一つの経路を選び出して中継を行う。これに対してDeleGateでは，アプリケーション層において中継を行うことで，中継するデータの内容や接続元（クライアント）に関して得られる情報を



(A) 実際のサーバ構成と要求／応答の流れ



(B) クライアントから見たサーバ構成

実際には(A)ような中継処理が行われているが，クライアントからは(B)の図のように見える。クライアントにはFirewall内部にあるホストの名前やIPアドレスは見えないので，セキュリティ上の安全性が増す。

図3 双方向の書き換えによるマウント機能の実現

[†21] この機能はその名の通り，ファイルシステム，特にネットワークファイルシステムのマウント機能を模したものである。

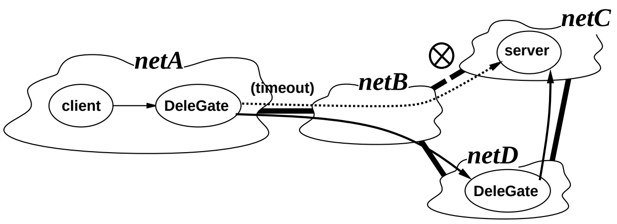


図 4 障害経路の自動的な迂回

利用した簡便で柔軟な経路制御を実現している。

ドメイン名による簡便な経路制御

多くの場合経路制御の単位は，組織体の階層を表現するドメイン名の単位に一致している。例えば国内において「*.etl.go.jp なら WIDE 経由で電総研へ」，あるいは海外から「*.ac.jp なら SINET 経由で日本へ」といった具合である。一般に，階層的な組織体とそれらが持つ IP アドレスの間の間のマッピングは複雑であるが，階層的なドメイン名のレベルでの経路制御では，その問題を回避することができる。

特に，中継するデータの中に目的のホストがドメイン名として書き込まれているアプリケーションプロトコルでは，文字列の単純なパターンマッチングによってこのような制御が効率的に行える。目的のホストが IP アドレスで直接指定してある場合にも，経路制御の効率は落ちるが，ドメインネームサーバ（DNS）を用いて IP アドレスからドメイン名を求めて経路制御に用いることができる。

障害経路の迂回と負荷分散

目的の情報への通常経路に障害があって通過できない場合に，自動的に別の経路を選択することができる。例えば，図 4のようなネットワーク構成において，client から server にアクセスする場合，通常は netA → netB → netC のように直接通信し，もし netB と netC の間が IP レベルで不通になった場合に，自動的に netD に置いた DeleGate に迂回して netC に中継させることができる。

一般に，このようなネットワーク構成で netB と netC の間の接続管理や障害回復に netA の管理者や利用者が関わる事はできない。しかし，DeleGate の利用によって容易にこのような状況に遠隔のネットから対処することができる。また DeleGate によるアプリケーションレベルの経路制御は，ネットワーク管理に関する権限を持たないエンドユーザからも利用することができる。

実際には遠隔地のネットワークの切断を直接知る事は容易では無いので，接続の手続きにタイムアウトを設けることによってこれを近似的に判断する。一定時間内に接続が成立しなかった時に，別経路による接続の試行に移るのである。従ってこれは，過負荷状態のネットワーク経路を自動的に迂回して負荷分散を実現する方法ともなっている。

アプリケーションレベルでの接続が失敗する原因として，目的ホストへの IP 的な到達性が無い (unreachable) こと以前に，ドメイン名として指定された目的ホストの名前を (DNS などにより) IP アドレスに変換することが出来ない (unresolvable) ことが少なくない (経験的にはむしろこちらのほうが多い)。ある DeleGate が，複数の DeleGate に向けた冗長経路を持つ場合，目的のホスト名から IP アドレスへの変換を行うことのできる DeleGate への経路が自動的に選ばれることになる。

DeleGate サーバ同志が多段に接続する際のプロトコル (DeleGate プロトコル) では第 2 版以降，先方の (目的サーバに近い)DeleGate が最終的に目的のホストとの接続を成功したか否かを知る機能が加えられている。これによって DeleGate が多段連鎖されている場合でも，上述のような自動経路選択が行われるようになった。また，このプロトコルを拡張して，先方の DeleGate 連鎖における負荷状況の情報を収集し，それをもとに経路制御を行う機能を実現することが可能である。

アプリケーションごとの経路制御

DeleGate では，目的サーバのドメイン名／アドレスによる以外にも，個々のアプリケーションプロトコルごとに，あるいはクライアント側のホストや利用者ごとに応じて，経路を制御することができる。また，アプリケーションプロトコル上で交わされるデータの識別名やデータの内容自身によって経路制御を行うこともできる。実際 3.5に述べたマウント機能は，アクセス対象の情報の識別名 (URL) によって経路を選択するための機能としても用いられる。

アプリケーション層での経路制御では，これら以外にも豊富な情報に基づく経路制御を実験し実現することが容易である。通信路に関して得られる様々な情報を反映した経路制御なども，最終的には専用のルータに組み込まれるとしても，その開発を支援するためのテストベッドとして，DeleGate を利用することができる。

3.7 コネクションキャッシュによる応答の高速化

広域ネットワークを利用するとき，クライアントの利用者は，データの転送に時間がかかる問題以前に，サーバへの接続そのものにかかる待ち時間を意識させられる。特にHTTPでは，一つの情報の取得ごとにこの接続作業が繰り返され[†22]，一方個々の情報のデータ量は小さいことが多いので，利用者の待ち時間は接続待ちの時間で支配されることになる。

さらに，FTPやNNTPのように本来は一つの接続で繰り返し情報を取得できるプロトコルに対しても，DeleGateやProxyのようなHTTPの中継サーバを介してアクセスすると，クライアントと中継サーバの間のHTTPプロトコルによる接続が情報の取得毎に作り直されるため，同様な問題が生じてしまう。すなわち，FTPサーバやNNTPサーバに対して，一つのファイルや記事の取得ごとに，接続，要求，応答，切断という作業が繰り返されてしまう。これは極めて不効率である。利用者に対する応答性が低下するばかりでなく，ネットワークや目的のサーバに余計な負荷をかけるという点でも問題である。

そこでDeleGateでは，一旦DeleGateが目的サーバとの接続を確立すると，その接続(connection)を一定時間保持して再利用する「コネクションキャッシュ」と名付けた方式を実現した。名前が示すようにこの方式は，データキャッシュがデータの保持・共有・再利用によってデータ転送に関するコストを削減するのと同様に，コネクションに関して同様な効果をもたらすことを意図したものである。いずれかのクライアントが，DeleGateを経由してある目的サーバに対して接続を行うと，そのコネクションは複数のクライアントから共有して再利用できるため，この方法は個々のクライアントが直接FTPサーバに接続するよりも効率が良い。

現在のところ，anonymous-FTPサーバに対するコネクションキャッシュが実現されている。例えば電総研からインターネットを通じて遠隔のFTPサーバに接続するのには，一般に数秒間かかるのが現状であるが，コネクションキャッシュによりこの待ち時間が無くなる。HTTPのクライアントを用いてFTPのディレクトリを表示しながら移動する場合，移動のたびに数秒要していたものが，ほとんどの場合一秒以内で移動できるようになった。

接続の作業には，サーバの起動，ユーザの認証とアクセス権のチェックなどが含まれており，その処理自体コストが高い。きめの細かいアクセス制御を行おうとするほどこの問題が大きくなるので，頻繁に接続作業が繰り返されることは，アクセス制御の高度化に対する障害となる。したがって，今後コネクションキャッシュの方式をより一般化して実現し，接続のコストを低減することで，接続に関する応答性だけでなく，セキュリティの向上にも寄与できると考えられる。

3.8 開発の状況

現在の最新版，DeleGate2.5.8は，約25000行のCプログラムにより実現されている。DeleGateは開発の初期(94年6月)からanonymous-FTPにより公開・配布されており，Appendix-Aに示すように現在までに国内外の600を越える組織がこれを取得している。各組織において利用されているだけでなく，全国の幾つかの大学，海外の大学，あるいは各ネットワークプロバイダ等から，不特定多数の利用者向けにDeleGate機能のサービスが行われている[†23]。第1版のDeleGateではHTTPの日本語コード変換機能のみが利用されていることが多かったが，第2版以降では，HTTP以外のプロトコルも含めてfirewallを越えて中継するためのシステムとしての利用が広まりつつある。

配布開始から1年間，活発な機能拡張および問題点の修理のための頻繁なバージョンアップが続けられており，現在までの配布ファイルの更新回数は168回を数える。公開後，障害の報告や改良に関する議論は，インターネット上に運営しているDeleGateのためのメーリングリストを通じて行われて来た。Appendix-Bに活動状況を示すように，このメーリングリストを通じた活発な議論を通じて，数多くの改良・改善がなされて来た。メーリングリストへの参加者の数も400名に達している。

DeleGate第1版では，HTTPの中継を中心とする基本的な中継機能を実現した。その後第2版では，機能面においてはアクセス制御機能の強化，3.4に述べたような既存クライアントのための中継機能の拡張，WAIS，CU-SeeMeなど中継するプロトコルの拡大が行われた。性能

[†22] HTTP/1.0の規格上では，一つの接続で複数の情報を取得する方法にも言及されているが6)，現時点では実用化されていない。

[†23] 主に文字コード変換機能の提供を目的とするものである。このサービスを最初に始めた東海インターネットワーク協議会の高木氏 <*takagi@center.nitech.ac.jp*> により「公開DeleGateサーバ一覧」が <http://www.tokai-ic.or.jp/WWW/delegate/> にまとめられている。

面では，コネクションキャッシュの導入，並列 DeleGate サーバの導入，および実装の改良により，応答速度およびスループットを飛躍的に向上させた。第 1 版の最終版（1.9.20）と現在の版を比較すると，応答速度において約 10 倍の高速化（軽量化）がなされている。現在の開発の方向は，firewall 用のシステムを越えて，次章 4.1 に述べるような機能により，アプリケーション層の仮想バックボーンネットワークシステム実現の模索に向かっている。

§ 4　DeleGate の今後

DeleGate 第 1 版および現在までの第 2 版の開発・試用・評価を踏まえて，今後，以下のような拡張，改良を行うことを検討している。

4.1　アプリケーション層の仮想バックボーン

4.1.1　汎用のトランスポート・チャネル

DeleGate 第 2 版では，目的サーバと DeleGate との間に動的に作られた接続を半固定的に維持する「コネクションキャッシュ」を導入した。現在のところこれは HTTP からの FTP サーバに対する接続にのみ有効であるが，このような機能は他のプロトコルでも必要とされている。

例えば，FTP プロトコルにおいては，要求や処理状態の交換を行う「コントロールコネクション」は固定的であるが，ファイルやディレクトリのデータ転送ごとに「データコネクション」を作成するようになっている†24。多段の DeleGate 間で FTP を中継する場合にも，このデータコネクションの作成の繰り返しによる遅延が起こる。また，データコネクションをサーバからクライアントに向けて接続するため，セキュリティ上の問題も生じる†25。そこで DeleGate 第 2 版では，FTP のデータを，要求や処理状態を中継するのと同一のコネクションの上で繰り返し転送する DeleGate 固有の転送モードを導入している。

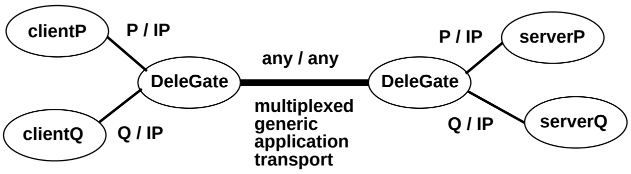


図 5　DeleGate トンネル

また，DeleGate 第 1 版では，多段の DeleGate 間を固定的な，IP に依存しない方法で接続する「トンネル」機能を試験的に導入した[1]。これは，サーバとクライアントとの間が IP 的に接続されていない，あるいは使用するアプリケーションプロトコルが途中経路を通過できない場合に，そのプロトコルを DeleGate サーバの連鎖によるトンネルを通じて通過させ，サーバとクライアントの間を中継するものである (図 5)。

また筆者は，複数のプロセス間でマルチキャストが可能な，VIABUS と呼ぶプロセス間メッセージ通信システムを開発した[3]。VIABUS においても，アプリケーション層のプログラムに対して，TCP/IP のような下層の具体的な通信手段を意識せずに通信が行える機能を提供することが目的の一つであった。

これらに共通して，TCP/IP や UDP/IP などの下位のレイヤの通信プロトコルから独立した，アプリケーション層のトランスポート手段が必要とされている。いわゆる遠隔手続き呼出し (RPC: remote procedure call) はそのような階層に属しているが，ここで求められているような要求に適合するものではない。必要なものは以下のような機能を持つ「汎用のアプリケーション・トランスポートチャネル」である。

- 同時に双方向の通信ができること (全二重)
- 同時に複数の通信ペアから利用できること (多重化)
- データの終了を確実に判定できること (異常終了の場合を含めて)

これは，機能的には単純なものであり，下位の通信レイヤに僅かの機能を加えただけで実現が可能である†26。

---

†24　FTP の規格上は，データ転送モードとして，一つのデータごとに接続を作る STREAM モードに対し，一つのデータコネクションの上で繰り返し複数のデータを送る BLOCK モードが定義されているが，BLOCK モードは実用化されていない。

†25　サーバからクライアントに向けて接続を行う PORT コマンドの場合。その逆である PASV コマンドではこの問題は無いが，このコマンドを実装していない FTP サーバもあり，未だ普及の途上である。

†26　このチャネルの実現のために，PPP(Point-to-Point Protocol)[18]を利用することを検討している。PPP は基本的にハードウェアによる通信路を想定して設計されたプロトコルであるが，これをソフトウェアレベルのプロセス間通信の手段として考えると，その応用は興味深い。特に，多対多のプロセス間通信を一本の論理的接続の上で中継する DeleGate のようなソフトウェアでは，その仕様を直接的に利用することができる。これと同様に SNMP のような，分散したハードウェアを制御する

アプリケーション層の各種トランスポートプロトコル（FTP, NNTP, SMTP, HTTP, など）において，実際のトランスポート制御部分は共通化できるはずである。これを共通化すれば，未だ実用化されていないHTTP上での複数の要求/応答の転送や，FTPのデータコネクション上の複数のデータの転送を共通に解決できる。また，運用においても実際に一つのトランスポートチャネルを共有して複数のプロトコルを同時に運ぶことができる。そして，この共通チャネルにアクセス制御機能を持たせれば，全プロトコルに共通のアクセス制御機構が提供される[†27]。

4.1.2　仮想バックボーン

広域ネットワーク上の分散した拠点にDeleGateを置き，DeleGate間を固定的に接続することにより，アプリケーション層の「仮想バックボーンネットワーク」を構築する。一旦このバックボーンに接続したアプリケーションの通信は，仮想バックボーンを構成する冗長な経路からDeleGateの経路制御に従い，最適経路で転送され，障害経路の自動的な迂回のサービスを受ける。各拠点に各アプリケーションプロトコルに対するキャッシュを置き（あるいはその近傍にあるキャッシュサーバやミラーサーバの情報を置き），最適経路はこのキャッシュの存在位置も考慮して決定する。

このようなバックボーンネットは，前節に述べたような「接続の繰り返し問題」を持つ，既存のクライアント/サーバ間の通信を高速化するためにも有用である。HTTPを例に取ると，クライアントはまず近傍のProxyサーバとしてのDeleGateに接続する。すると，そのDeleGateは遠隔にある目的のHTTPサーバの近傍にあるDeleGateへの経路を選び，クライアントからの要求を転送する。そして，遠隔のDeleGateが目的のHTTPサーバと接続する（図5と同じ接続関係）。クライアントとその近傍のDeleGate，サーバとその近傍のDeleGateとの通信は従来型のプロトコルで行われるが，近傍であるために接続の繰り返しによるオーバヘッドは小さい。一方，その間を中継するDeleGate間の通信では，固定的なトランスポートチャネルにより接続の繰り返しのオーバヘッドは無い。

4.1.3　クライアントとDeleGate間の固定接続

ProxyクライアントやSOCKSクライアントの場合，通信を行うためには必ず同一の中継サーバに接続することになる。にも関わらず，クライアントからの要求毎に毎回中継サーバに接続し直すのは無駄である。もし，クライアント/中継サーバ(DeleGate)の間に固定的な接続を作り，それを繰り返し再利用できるようにすれば，このようなオーバヘッドを無くすことができる。これは，前述のような汎用のトランスポートチャネルを用いて実現が可能と考えられる。

4.2　キャッシュ制御

4.2.1　キャッシュ容量の制御

現在の版では，DeleGate自身にはキャッシュ量の上限を押える機能は無い。現状では，別プログラムにより，定期的に(例えば一日に一度)全キャッシュファイルの作成あるいは最終アクセスの日付・時刻を調べて，保存期限を越えたものを削除することで対処している。しかしこれは，キャッシュ量を小さく押えたい場合にそれが難しく，逆に巨大なキャッシュを許す場合に削除作業が集中・長時間化するという問題が起こる[†28]。このため，DeleGate自身がキャッシュ量を制限し逐次的に削除する機能が必要であり，今後の課題の一つである。

4.2.2　キャッシュ対象プロトコルの拡大

現在の版ではHTTPとGopherのみを扱っているが，同一データが繰り返しアクセスされ得るプロトコル(FTP, NNTP, WAISなど)全般に適用範囲を拡張する。現状においてネットワークニュースは，分散した自律的なサーバが各々に独自の管理をしなければならない状況であり，維持のコストが高い。DeleGateにNNTPの通信をキャッシュする機能を加えた上で，一つのニュースサーバを複数のDeleGateを窓口として共有するようにすれば，管理の手間のかかるニュースサーバの数が削減できるとともに，一つのニュースサーバに記事の取り出しや検索のための負荷が集中する問題を解決することができると考える。

4.2.3　自動更新とミラー

ある情報のキャッシュの更新は，ユーザによるその情報へのアクセスをきっかけとして行われる。この方法には，

ためのプロトコルを，複数のDeleGateサーバ間での実行監視制御に応用することも考えられる。

[†27] TCP/IP上のプロトコルに対して，外付けで共通のアクセス制御機能を付加するものとしてtcp_wrapperがある[20)]

[†28] 一般にスプール量が巨大になるネットワークニュースシステムなどでも，この問題に悩まされることが多い。

運悪くキャッシュの更新に出会ってしまったユーザにとっては救いにならないという欠点がある。例えば定期的に更新を行う，あるいは頻繁にアクセスされている情報は期限切れ直前に自動的に更新をかけるなどの対策が必要である。この機能は，httpd など他のキャッシュ付き proxy でも共通に利用可能なものとして，DeleGate の外部の機能として実現すべきかもしれない。このような自動更新の機能は同時に，他のサーバの情報を複製して代理サービスする，いわゆる「ミラーサーバ」の実現の基礎となる。

4.3　アクセス制御の高度化

各プロトコルに含まれる個々のコマンドの単位でのアクセス制御を導入する必要がある。また，既存のプロトコル (NNTP, SOCKS, HTTP, ...) を中継するために，従来それらが用いて来たアクセス制御ファイルの記述を取り込むためのパーサの生成系，そして，多様なアクセス制御を処理できる「汎用アクセス制御エンジン」が望まれる。

4.4　複数 DeleGate サーバの統括管理

複数の DeleGate サーバを協調的に動作させたり共通の環境を持たせるために，複数の DeleGate サーバの起動と接続，制御を統括して行う必要がある。また，より複雑なアクセス制御を記述する一方，記述を容易にすることも要求されている。これらを実現するために，configuration ファイル，管理用データベース，統括制御用サーバなどを導入する必要がある。

4.5　コネクション指向の CGI

DeleGate サーバ（プロセス）に対して他のサーバ（プロセス）から機能を付加できるようにするために，継続的なプロセス間の接続を通じて機能提供を行う際共通のインターフェイスである。HTTP においては，一つの要求に対して外付けのコマンドを起動して応答を生成する CGI (Common Gateway Interface) があるが，そのコネクション指向版とも言える。現在の DeleGate の実装では，一つのプロセスの中に全ての機能が組み込まれているが，このようなインターフェイスの導入により，効率を落とすことなく DeleGate をモジュール化することで，今後の拡張やカスタマイズを支援する。実際には DeleGate 同志の間の通信プロトコル (DeleGate プロトコル) と共通になることも考えられる。

4.6　その他

以上の他にも，今後の DeleGate で実験，実現を試みられるべき多くの課題がある。その一部として以下のようなものがある。

MIME と HTML の変換: インライン・イメージを含む HTML ページと，MIME の *multipart* メッセージとの双方向変換。これにより，WWW の任意のページを一つのメールにして送信したり，逆にメールを送って WWW のページを更新することが容易になる。このところ，MIME 対応のメール操作ツールが普及し始めており，この変換機能の需要は今後高まると思われる。

経路制御のためのプロトコル: DeleGate 間のネットワークとして構成される仮想バックボーンにおける経路制御プロトコル。ルータ間のプロトコルである Border Gateway Protocol[17)]の利用も検討したい。

マウント機能の一般化: FTP や Gopher プロトコルへの適用。これを自動更新付きのキャッシュと組み合わせることで「ミラーサーバ」が実現できる。また，複数の NNTP サーバの統合 (ニュースグループの併合) にも利用できるようマウント機能の一般化を考える。

§ 5　おわりに

開発中のプロトコル中継システム DeleGate について，その開発の背景と実現した機能の概要，および今後の展望を述べた。DeleGate は，防火壁や文字コードの問題など，現状において差し迫られた問題を解決する実用システムとしての役割を果たしている。その一方で，従来のクライアント / サーバモデルの中間に中継サーバという段階を導入することにより，様々な付加価値機能や性能の向上を，クライアント / サーバのいずれにも変更を加えることなくもたらせる可能性を示している。

現在の DeleGate は，実用システムとしてどのような機能が必要であるか，またどのような新たな機能が実現可能であるかを探りつつ手早くプロトタイピングを繰り返して改良されている段階である。次の段階ではこの経験を糧に，本質的に必要なシステムの基本機能を抽出し，再構成を行いたい。その枠組には，汎用のパターンマッチングと書き換えの機構，そして並列計算モデルに基づく言語・システムが重要な位置を占めるものと思われる。

今後も，実用システムとしての完成を目指すとともに，

新たな分散情報システムの機能と構成方式への手掛かりを得る探針として，DeleGateの開発と活用を継続して行きたい。

謝辞

本研究の機会を与えていただいている情報アーキテクチャ部太田公廣部長，同部情報ベース研究室大蒔和仁室長に感謝いたします。本システムを開発する動機となった電総研のWIDEインターネットへの接続と「出島方式」は言語システム研究室戸村哲室長をはじめとする電総研‘IPマスター’の方々の御尽力により維持されて来ました。工技院ネットワークを経由したTISNインターネットの利用，およびその防火壁上でのDeleGateの運用にあたっては，工技院情報計算センター(RIPS)の瀬河浩司氏(現 情報ベース研究室)に多くのご協力をいただきました。計算機方式研究室の山名早人氏のまとめられたDeleGateをパーソナルコンピュータから利用するための手引は，多くのユーザをWWWやGopherの世界に誘い出し，DeleGateの普及を促進しました。そして，DeleGateメーリングリストの参加者の皆様はじめ，百人を越える方々から電子メールや電子ニュースを通じてDeleGateの改良に有益なコメントを頂きました[†29]。以上の方々に感謝いたします。

工業技術院職務作成プログラム等管理番号

1994-ETL-E8715-01

参考文献

1) 佐藤豊: プロトコル中継システムDeleGateの開発, 電総研研究速報 TR-94-17, (1994).
<URL:ftp://etlport.etl.go.jp/pub/DeleGate/ETL-TR-9417.ps.gz>

2) 佐藤豊: DeleGateに見るインターネット上の公共ソフトウェアの成長過程, 機械振興 **28**, **4**, (1995) 96-101.

3) 佐藤豊: 分散オブジェクト指向UIMSの実行時アーキテクチャの設計と実現, 電子技術総合研究所彙報 **58**, **10**, (1995) 63-71.

4) 田代秀一: インターネットの現状と課題, 電子技術総合研究所彙報 **58**, **5**, (1994) 21-33.

5) 高田敏弘: World-Wide Webの日本語化・国際化の現状, 日本インターネット協会ニュース **1**, **2**, (1994).
<http://www.ntt.jp/people/takada/docs/www-j10n/>

6) T. Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen: Hypertext Transfer Protocol – HTTP/1.0, Internet-Draft, (1995).
<http://www.w3.org/hypertext/WWW/Protocols/Overview.html>

7) A. Luotonen and K. Altis: World-Wide Web Proxies, WWW'94 Conference, (1994).
<http://www.w3.org/hypertext/WWW/Daemon/User/Proxies/Proxies.html>

8) T. Berners-Lee, L. Masinter, M. McCahill: Uniform Resource Locators (URL), A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network, RFC1738, (1994).
<ftp://ds.internic.net/rfc/rfc1738.txt>

9) T. Berners-Lee, D. Connolly: Hypertext Markup Language - 2.0, Internet-Draft, (1995).
<ftp://ds.internic.net/internet-drafts/draft-ietf-html-spec-03.txt>

10) F.Anklesaria et al., The Internet Gopher Protocol, Internet RFC 1436, (1993).
<ftp://ds.internic.net/rfc/rfc1436.txt>
<gopher://boombox.micro.umn.edu:70/11/gopher/gopher_protocol>

11) N. Borenstein and N. Freed: MIME (Multipurpose Internet Mail Extensions),
Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC1521, (1993).
<ftp://ds.internic.net/rfc/rfc1521.txt>
K. Moore: MIME Part Two: Message Header Extensions for Non-ASCII Text, RFC1522, (1993).
<ftp://ds.internic.net/rfc/rfc1522.txt>

12) Y.Rekhter, B.Moskowitz, D.Karrenberg and G.de Groot: Address Allocation for Private Internets, RFC1597, (1994).
<ftp://ds.internic.net/rfc/rfc1597.txt>

13) J. C. Mogul: Simple and Flexible Datagram Access Controls for Unix-based Gateways, USENIX Technical Conference, (1989).
<ftp://gatekeeper.dec.com/pub/misc/vixie/>

14) D. Koblas and M. R. Koblas: SOCKS, USENIX Security Symposium III, (1992).
<ftp://ftp.nec.com/pub/security/socks.cstc/>

15) M. St. Johns: Identification Protocol, RFC1413, (1993).
<ftp://ds.internic.net/rfc/rfc1413.txt>

16) J. Murai, M. Crispin and E. van der Poel: Japanese Character Encoding for Internet Messages, RFC1468, (1993).
<ftp://ds.internic.net/rfc/rfc1468.txt>

17) Y. Rekhter and T. Li: A Border Gateway Protocol 4 (BGP-4), RFC1654, (1994).
<ftp://ds.internic.net/rfc/rfc1654.txt>

18) W. Simpson: The Point-to-Point Protocol (PPP), RFC1661, (1994).
<ftp://ds.internic.net/rfc/rfc1661.txt>

19) T. Fitzgerald: UDP packet relayer, (1993).
<ftp://ftp.wang.com/pub/fitz/>

20) W.Z. Venema: TCP WRAPPER, network monitoring, access control and booby traps, UNIX Security Symposium III Proceedings (Baltimore), (1992).
<ftp://ftp.win.tue.nl/pub/security/tcp_wrapper.ps.Z>

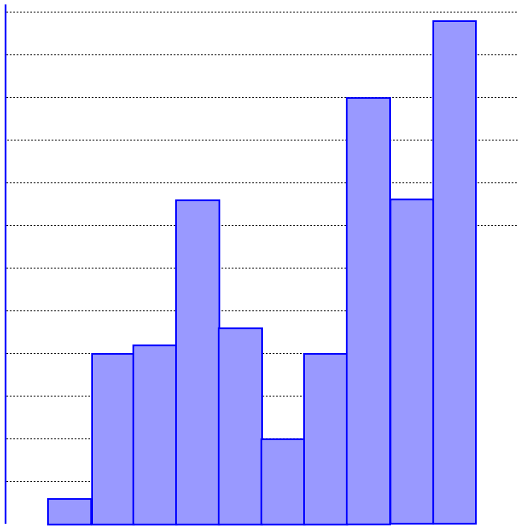
[†29] DeleGate開発協力者リスト: <URL:ftp://etlport.etl.go.jp/pub/DeleGate/CREDITS>

Appendix-A: DeleGate の配布状況

DeleGate の最新版は，anonymous-FTP により以下のディレクトリから配布されている。

$< URL : ftp : //etlport.etl.go.jp/pub/DeleGate/ >$

このディレクトリに置かれた DeleGate 配布ファイル (delegate*.tar.gz) に対する anonymous FTP による転送状況の統計を以下に示す (94 年 6/12 配布開始 ~ 95 年 5/31 現在まで)



配布先（606 機関）の内訳は以下のとおり。

国内:	499
大学・研究機関	210
民間企業	191
政府関係	36
ネットワーク組織	19
その他の団体	43

海外:	107
米国企業 (*)	58
米国大学研究機関	20
米国政府	1
その他米国の組織	1

カナダ	2
フランス	2
ドイツ	3
イギリス	2
ギリシャ	1
ブルガリア	1
オーストラリア	4
インドネシア	1
インド	1
韓国	1
台湾	1
ネットワーク組織	8

(*) 在日の米国系企業を含む

anonymous-FTP による配布の他にも，フリーソフトウェアを集めた幾つかの CD-ROM に収録され配布されている。

インターネットマガジン, 付録 CD-ROM (94/09 ~ 各号)
日本 DEC, DEC ALPHA 用フリーソフト '94 (94/09)
日本 SUN ユーザーズグループ (NSUG), '94 (94/09)
日本 UNIX ユーザ会 (JUS), '94 (94/10)

Appendix-B: DeleGate 開発への参加・協力

DeleGate に関するバージョンアップ情報の通知，障害報告，改良のための議論などは，DeleGate メーリングリスト

$< delegate@etl.go.jp >$

で行われている。以下に，この ML の活動状況を示す。

著 者 紹 介

佐藤 豊
Yutaka Sato
情報アーキテクチャ部情報ベース研究室
ysato@etl.go.jp

DeleGate 利用者インターフェイス管理システム，ネットワーク指向ソフトウェア構成法の研究に従事。