電総研研究速報

TR-94-17

プロトコル中継システム **DeleGate** の開発

Development of a Protocol Mediation System *DeleGate*

佐藤　豊

Yutaka Sato <ysato@etl.go.jp>

情報アーキテクチャ部 情報ベース研究室

受付日: 1994 年 11 月 15 日

インターネット上のプロトコル (通信手順) を中継するシステムとして開発中の *DeleGate* について報告する。近年、実用ネットワークとして全世界的に規模を拡大しているインターネットであるが、現状では技術面や整備の面で残された問題は少なくない。例えば、セキュリティ保護のために設けられる防火壁 (*firewall*) と呼ばれる機構がせっかくの豊富な提供サービスへのアクセスを阻害し、インターネットへの低速な接続回線がマルチメディア情報へのアクセス時間を耐え難いものにし、日本語の文字コードの不統一のためにせっかくアクセスできた情報が表示できない、などの問題である。より多くの利用者による活用を促進するために、これらの問題の解決が望まれている。 DeleGate は、防火壁の上で各種のプロトコルを自動中継する機構を基礎として、これらの問題を解決した。通信の内容を双方向に、すなわち要求内容だけでなく応答内容も解釈しつつ中継する点が DeleGate の特徴である。これにより、クライアント / サーバシステムのどちらにも変更を加えることなく、上述の諸問題を解決するとともに、クライアントに対して新たな機能を提供することを可能にしている。開発開始後 8ヵ月、試用評価のための公開後 5ヵ月を経た現在、 DeleGate は国内外の 300 を越える組織で試用され、その評価を反映しつつ改良が続けられている。本論文では、 DeleGate 第 1 版の開発の経緯と実現の概要について報告する。

電子技術総合研究所

〒 305 茨城県つくば市梅園 1-1-4

Phone: 0298-54-5061

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、その半年後の 7 月には 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、企業から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。これにともない、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 [1]や Gopher などの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。

このような大規模化、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、緊急に解決しなければならない以下のような問題点に直面している。

- セキュリティの問題: セキュリティ保護はますます重要になる。しかし、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」を設けると、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: 個人利用者などが低速安価な回線で接続し始めている。しかし通信回線が遅いと、特に画像や音声などの大きいデータの転送が、実用に耐える速度で行えない。
- 文字コードの問題: パソコン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[2]。このため、サーバの提供する情報の文字コードと、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[3]。

これらの問題の解決に迫られて、 DeleGate の開発は始められた [4]。 DeleGate は防火壁の上で動作して、防火壁によって遮断されているインターネットプロトコル (IP) を、アプリケーション層で中継するソフトウェアである。 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、さらに必要に応じて柔軟に機能を拡張することもできる。例えば、アプリケーション層でのプロトコル中継により、以下のような機能拡張が容易にできる。

- 非 **IP** による中継: IP でないプロトコル、たとえば低速なシリアル回線や高速な ATM などを有効に利用できるプロトコルの上で、各種の IP プロトコルを中継する。
- 賢い経路選択: IP ルータのレベルでは実現不可能な、アプリケーション・プロトコル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を実現する。

DeleGate の開発は 94 年の 3 月に始められ、同時に電子技術総合研究所 (以下、電総研) 内で試用に供された。その後工業技術院 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 (anonyumous FTP) での配布を開始、その後

---

[1]スイスの CERN で始められたハイパーテキストベースの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 (Hyper Text Transfer Protocol) を用い、 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 形式を採用している。ハイパーテキストの記述には SGML をベースとする HTML(Hyper Text Markup Language) を用いる。HTML 中で他の情報を指すのに使われる識別名としては URL(Uniform Resource Locator)[URL] が用いられる。クライアントプログラムとしてはイリノイ大学 NCSA で作成された Mosaic が有名。

[2]この問題は本来、文字コードの統一化で解決されるべきだが、それぞれの文字コードによるデータやプログラムの膨大な資産があるため、統一化は非常に困難に思われる。

[3]現在のところ、各組織における DeleGate の使用目的で最も多いものは、日本語コード変換であろう。

[4]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。

幾つかのフリーソフトウェア集 CD-ROM に収められて再配布されている。同年 11 月現在、 DeleGate は国内外の 300 を越える組織で試用されており、試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。以下では、上述の問題点の解決や可能性の実現のために、DeleGate 第 1 版で実現した方式の概要と、第 2 版以降への展望を述べる。

2 防火壁を越えて – 内から外へ –

2.1 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[5]。これは不便であるばかりでなく、本来利用可能であるはずのサービスやユーザインターフェイスが使用できないことになってしまう[6]。

2.2 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、アプリケーション層の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。

- 汎用プロトコル中継システム
  代表的なシステムとして *SOCKS* が広く使われている [SOCKS]。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[7]。 Unix 用には、 SOCKS 関数ライブラリが配布されて、 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はあまり普及していない。

- 個別プロトコル中継システム[8]
  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* [PROXY] が広く使われている (以下これを *Proxy* と呼ぶ)。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対象を示すのに用いられる URL で各種のプ

[5]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。
[6]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。
[7]Rconnect,Rgetsockname,Rbind,Raccept,Rlisten,Rselect の各関数
[8]SRA の歌代氏 <*utashiro@sra.co.jp*> は、 Gopher プロトコル中継用の *GopherGate* を開発した。

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) 全ての要求を 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完全な URL を送る、という変更を加えてやれば良い。

2.3 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、 1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、 CERN-httpd でとられた方式は汎用のように見えるが、その適用範囲は以下のように限られており、 SOCKS のような汎用の中継システムに比べてカバーできる範囲が狭い。

- telnet などのようにクライアント / サーバ間の継続的な接続を前提としたコネクション指向の *stateful* なプロトコル[9]には、適用できない。
- 最初にクライアントからサーバに対して、アクセスしたいリソースの識別名を要求として送るプロトコルにしか、適用できない。

1章に述べた問題は、 CERN-httpd 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、これらを総合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。これはまた、 stateless なプロトコルを複数統合する Mosaic などの WWW クライアントと同様な、 stateful な複数のプロトコルを統合したクライアントの実現を支援する機能とも成り得る。

DeleGate は、 SOCKS のような汎用の中継システムとしても、 CERN-httpd 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[10]。現在中継が可能なプロトコルは、 FTP, Telnet, X, SMTP, POP, NNTP, Whois, Finger, Gopher および HTTP である。中継システムとしての DeleGate は以下のような特徴を持っている。

- stateful なプロトコルも中継可能
- 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
- CERN の Proxy-httpd と互換 (Proxy の代わりとして利用可能)
- Proxy に非対応のクライアント、あるいは Gopher 専用クライアントも中継可能

最後の点は、 DeleGate が Proxy と異なり、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、後述 (5章) のマウント機能も、これによって実現されている[11]。

---

[9] 継続的に接続されたクライアントとサーバの間で、過去の交信に依存する状態を保持し、その状態に依って交信の内容が解釈されるプロトコル。一方、 HTTP や Gopher は、一度の接続あたり、一つの要求に対して一つの応答を得て接続を終了する、 stateless なプロトコルである。

[10] 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。

[11] DeleGate の開発では最初に、 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。

2.4 ルータによる解決法との比較

従来セキュリティ保護は、アプリケーション層ではなく、 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一般の 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 [SCREEND]。

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。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アドレス [RFC1597] 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 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、以降の章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 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、日本国内では未だに 64Kbps の場合が多い。また、最近増え始めた個人利用者の接続の場合には、音声回線上の PPP (Point-to-Point Protocol) を用いた 38Kbps が主流であろう。このような低速な回線では、実時間性への要求が低い電子メイルなどの応用は別として、対話的な双方向通信を必要とする WWW などの本格的な使用には耐えない。 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では十分な応答性が得られない。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 の回線では最短でも 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[12]。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[13]。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[14]。

[12] 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

[13] 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

[14] 現在の LAN の速度は 10Mbps ~ 100Mbps が一般的である。クライアントが DeleGate サーバと同一マシンにあれば、ディスクの読み出し速度、場合によってはメモリ読み出しの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。

表 1. **HTTP 用 DeleGate の利用状況とキャッシュの効果**
電総研において 1994 年 10/31(月) ~ 11/2(水) の間測定

	10/31		11/1		11/2	
a. クライアントホスト数	69		75		88	
b. 総アクセス回数	7103		6867		11369	
c. ローカルホスト参照 (*1)	1450		1198		1554	
d. DeleGate 内部データ参照 (*2)	96		211		245	
e. エラー	40		38		308	
f. キャッシュ無視 (*3)	738		361		1205	
g. キャッシュ参照	4779		5059		8059	
h. キャッシュミス	1763		1949		3334	
i. キャッシュヒット (*4)	3016		3110		4725	
m. 有効キャッシュヒット m/g%	2819	59%	2948	58%	4358	54%
j. 有効期限内 (*5)	2207		2420		3644	
k. 有効期限切れ後無変更 (*6)	612		528		714	
l. 有効期限切れ後変更	197		162		367	

[b = c + d + e + f + g , g = h + i , i = m + l , m = i + k]
(*1) 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。
(*2) DeleGate サーバ内に組み込まれているアイコンデータなどへのアクセス。
(*3) 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
(*4) キャッシュ保存期限: 30 日
(*5) キャッシュ有効期限: 24 時間
(*6) 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が再利用される。

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[15]。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。例えば数秒程度、場合によっては数十秒を要することも稀ではない。また、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 1 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[16]。全アクセス数 b から、キャッシュを利用しない場合 (c ~ f) を除いた数が、実際のキャッシュへの参照数 g 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 i であるが、有効期限切れかつ元データも更新されていた場合 l は無効である。結局、有効なキャッシュヒット数 m を、キャッシュ参照回数 g と比較すると、 50 ~ 60% のヒット率である。このようにキャッシュによって、リモートのサーバとの通信が半減されている。

[15] 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。

[16] 利用者の数を調べることは現在の DeleGate ではできないが、このホスト数を利用者数の下限として見積もって良いだろう。

表 2. キャッシュのためのディスク消費量
電総研において 1994 年 11 月 3 日測定　(キャッシュ保存期限: 30 日)

プロトコル	ホスト数 (*1)	ディレクトリ数	ファイル数	ファイル使用量
HTTP	1128	6489	15840	229Mbytes
GOPHER	496	3326	5095	241Mbytes

(*1) サーバホスト数。非標準ポートの使用による重複したカウントを含む。

表 2 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 460M 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、ほとんど無視できるコストと言えよう。

4　文字コードの適応

4.1　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、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に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[17]。

- 7 ビット JIS(JIS): インターネットでの公式名は ISO-2022-JP [ISO-2022-JP]。 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(または UJIS): 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が並立している現状は不幸であり[18]、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いる。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。そして、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[19][20]。

[17] 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。

[18] これらのうちで ISO-2022-JP は、国際規格 (ISO-2022) に基づき、インターネットでも IANA に公式に登録されており、多国語化に対応でき、様々なプロトコルによる転送において最も安全な表現形式でもある。筆者は ISO-2022-JP 以外の日本語文字コードは、インターネット上に流れて欲しくないと考える。

[19] そもそも、現状ではほとんどのネットワークソフトウェアは外国産であり、もともと日本語には対応していない。そのうえ内部表現への変換処理を要する ISO-2022-JP の場合、対応がオリジナルの段階で行われていることは望むべくもない。ましてや、3 種類の日本語コードがオリジナルの段階でサポートされることなどは、まず考えられない。

[20] 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、各種日本語コードだけでなく各国の文字コードに対応可能な拡張を行ったもの [L10N] が主に使われて来た。

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の統一は容易ではない。そして、現在の 3 種類のコードに加えて、 UNICODE あるいは ISO-10646 が加わろうとしている。

4.2 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[21] [22]。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。

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[23]。

(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[24]。

5 防火壁を越えて – 外から内へ –

2章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中継システムを導入した。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[25]。

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）ホストにある情報が、利用者から見ると全て DeleGate サーバのホストにあるかのように見せる技術である[26]。

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

[21] コード変換機能は、電総研所内のローカルニュースグループで、 Macintosh 上の Mosaic ユーザから JIS コードのテキストを読みたいという要求が出されたのを受けて追加された。結局はこの機能によって DeleGate が広く使われるようになった。

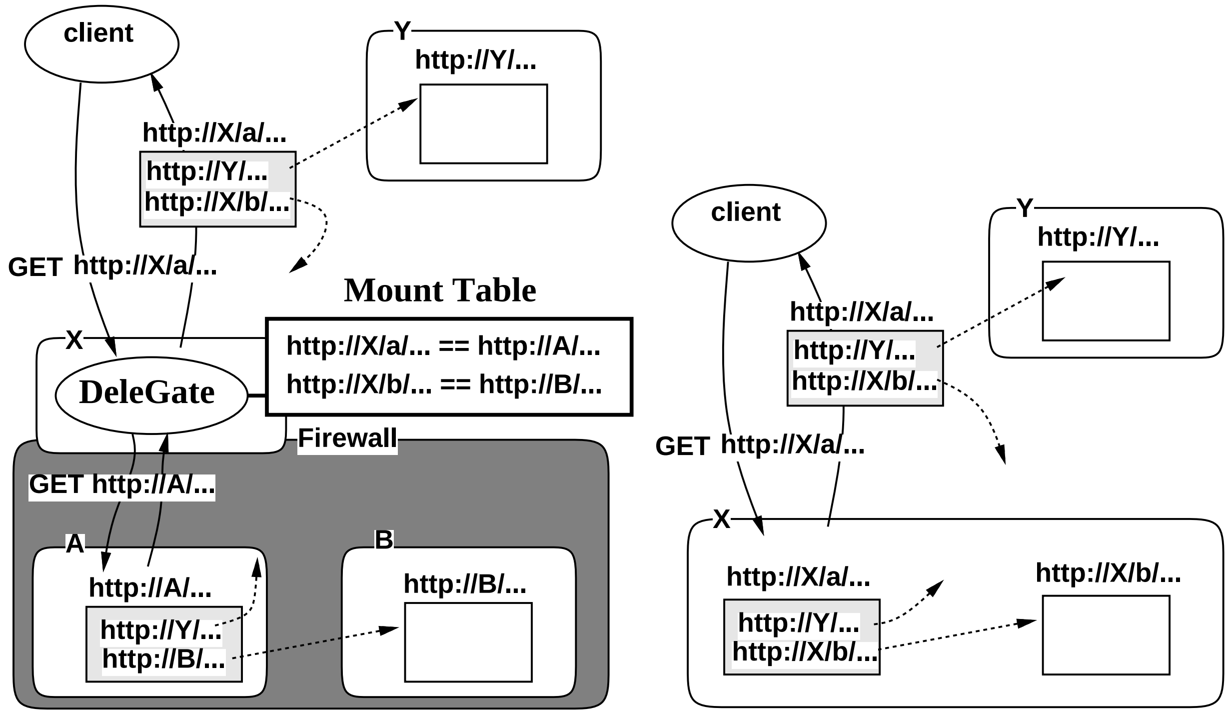
[22] DeleGate の他にも、 Proxy-httpd と併用する日本語コード変換サーバとして、奈良先端科学技術大学院大学の知念氏 <*k-chinen@is.aist-nara.ac.jp*> による *sjis − conv* が知られている。

[23] 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。

[24] 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。

[25] 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。

[26] この機能は、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に着想を得たものである。



実際には左の図ような中継処理が行われているが、クライアントからは右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 1: 双方向の書き換えによるマウント機能の実現

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。 DeleGate 第 1 版で実現されている、 HTTP サーバのマウントの例を図 1に示す。

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する一方、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、第 8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。

6　多重通信経路の選択

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、各経路への負荷の分散ができることが望まれる。また、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は、別の経路を選択できることも望ましい。多重の経路ははまず、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。

一般には、このような複数の経路の中から一つの経路を選び出すのは、IP 層のルータである。しかしルータには、例えば各経路の負荷状況を反映して経路を制御するような機能は無い。これに対してアプリケーション層のプロトコル中継システムでは、通信路に関する様々な情報を反映した制御を行うことができる。さらに、通信路以外の情報も利用することも可能である。例えば、より遠隔地にあるサーバの内容を複製してサービスする caching proxy や「ミラーサーバ」が近くにあることがわかっていれば、それを利用するよう制御することができる。

このため DeleGate では、目的の情報が置かれているホストのドメイン名やネットワークアドレスに従って、経路を選択することができるようになっている。具体的な制御方法については、次章に述べる。

7 DeleGate 第 1 版の概要

DeleGate1.9.18 は、約 12000 行の C プログラムで実装されている。DeleGate サーバは delegated という名前のプログラムである。また、ネームサーバ対応のものは delegatedR、SOCKS 対応のものは delegatedS という別々のプログラムになっており、利用者は利用目的や利用環境に応じてこれらを使い分ける。以下では、「DeleGate サーバ」を略して delegated と呼ぶ。

起動時のパラメタ

delegated の起動は、以下のように行う。

```
% delegated   -P ポート番号  パラメタ名 1= 値 1  パラメタ名 2= 値 2  ...
```

パラメタとしては以下のものがある[27]。(‘*’ を先頭に付けたパラメタは、同名のパラメタが繰り返し与えられても良いものである)

```
 SERVER    proto://host:port      -- この delegated で中継するプロトコル
*MASTER    host:port:HostList     -- マスターとなる汎用 delegated の在所
 PROXY     host:port              -- マスターとなる Proxy-httpd の在所
 TUNNEL    type:file-name         -- トンネルを開通させる SHIO スクリプトの名前
 FORCEON   ProtoList              -- 全ての要求を無条件に MASTER(PROXY,TUNNEL) に送る
 DELEGATE  host:port:ProtoList    -- URL や DirEnt の指す先をどこへ向け替えるか
*HOSTS     HostAddrList           -- ホスト名 /IP アドレスの対応リスト
*MOUNT     "path URL"             -- URL を http://${DELEGATE}/path にマウントする
*RELIABLE  HostList               -- 指定されたホストからの要求のみを受け付ける
*REACHABLE HostList               -- 指定されたホストへの中継のみを受け付ける
 RESTRICT  ProtoList              -- アクセス制限されるべきプロトコルのリスト
 CHARSET   {JIS,EUC,SJIS}         -- 応答テキスト中の日本語コードを何に変換するか
 CACHEDIR  dir-name               -- HTTP および Gopher のキャッシュの在所
 EXPIRE    {days,hours,secs}      -- キャッシュデータの有効期限
 LOGDIR    dir-name               -- ログファイルの在所ディレクトリ
 LOGFILE   file-name              -- ログのファイルの名前
 PROTOLOG  file-name              -- httpd 互換のログファイルの名前
 PIDFILE   file-name              -- delegated プロセスのプロセス ID の保存ファイル
 WORKDIR   dir-name               -- 作業ディレクトリ
 OWNER     user-name              -- 誰の権限で delegated を実行するか
 MANAGER   user-address           -- この delegated の管理者の電子メイルアドレス
 HTMLCONV  deent,enent            -- HTML 予約文字のエンコード / デコード
```

[27] これらのパラメタは、コマンド引数としても環境変数としても与えることができる。同一の名前のパラメタが環境変数とコマンド引数の両方で与えられた場合には、コマンド引数で与えられたものが優先する。

ホストおよびネットワークの表現

HostList は、ホスト (ネットワーク) の名前または IP アドレスを ',' で区切って並べたものである。以下に HostList の一例を示す。

```
hostA,192.31.200.10,hostB/255.255.255.0,192.168.[0-255],*.ac.jp
```

ネットワークは、そのネットワーク上にあるホストの名前または IP アドレスに「ネットワークマスク」をかけて表現する。上の例で hostB/255.255.255.0 がその例である。複数のホストまたはネットワークを表す方法としては、IP アドレスの範囲を [m-n] のように表現する方法と、'*' を用いてホスト名. ドメイン名の文字列のパターンを表現する方法がある[28]。

HostList は左から右に走査される。特定のホストを除外する場合にはそのホストの前に '!' を前置する。例えば、

```
*.jp,!*.co.jp
```

は「*.jp にマッチするもの , ただし *.co.jp にマッチするものを除く」を表す。また、以下の 2 行は同じ意味である。

```
1.2.3.[0-255],!1.2.3.50,!1.2.3.60
1.2.3.[0-49],1.2.3.[51-59],1.2.3.[61-255]
```

Specialist と Generalist

「SERVER=proto://host:port」が指定された場合には、この delegated は指定されたプロトコル proto 専用の中継サーバとなる。クライアントからの特別な指示がなければ、この delegated はクライアントから見て、host 上の port で proto プロトコルのサービスしているサーバのように見える。クライアントと交信するプロトコルは、proto で指定されたものに固定される。一方、SERVER が指定されなかった場合には、潜在的に全てのプロトコルを中継可能な汎用の中継サーバとなる。前者を *Specialist*、後者を *Generalist* と呼んで区別する。第 1 版では基本的に、キャッシュを行うのは Generalist の役割であり、文字コードの変換やプロトコル変換を行うのは Specialist の役割である。

DeleGate プロトコル

Generalist-delegated がクライアント[29]と交信するプロトコルが「DeleGate プロトコル」である。第 1 版の DeleGate プロトコルは、以下のようになっている。

(1) クライアントから delegated に接続する
(2) クライアントから delegated に、使用プロトコルや目的サーバなどの情報を送る
(3) delegated から目的サーバに接続する
(4) delegated は、目的サーバとクライアントとの間の通信を双方向に中継する

Generalist は、クライアントと目的サーバとの間の通信を、プロトコルの変換を行うことなくそのまま中継し、中継に伴ってキャッシュを行う。

多段中継時の中継先の指定

delegated は多段階に連鎖させることができ、ある delegated の先にある delegated を指定するパラメタとして、MASTER, TUNNLEL, および PROXY がある。連鎖した delegated の間では

[28] '*' は任意の文字列パターンにマッチする。
[29] Generalist のクライアントは、一般に delegated である。

以下のように必要に応じて中継が行われる。まず、指定された目的サーバのドメイン名が現在のdelegatedで解釈可能ならば、そのdelegated自身が目的サーバに接続を試みる。接続が成功したならば、当然その先のdelegatedへの中継は行われない。

一方、名前の解釈か接続が失敗した場合には、 MASTER, TUNNEL, あるいはPROXYで指定された先に中継する。中継先にあるサーバは、 MASTER, TUNNELの場合にはdelegated (Generalist)、 PROXYの場合にはHTTP専用delegatedまたはProxy-httpdである。

あるdelegated自身では目的サーバに接続を試みることなく、必ず中継先に強制的に中継してしまうことが必要になることがある。そのような場合には、 FORCEONパラメタに、強制的に中継すべきプロトコルの名前を列挙する。第1版では、 MASTER, TUNNEL, PROXYのいずれかが指定された場合には、デフォルトでFORCEONが適用され、必要な場合に FORCEON を解除する。

キャッシュの管理

第1版では、キャッシュは基本的に Generalist が管理する。そのため、キャッシュを利用するには、 Specialist だけでなく Generalist も走らせる必要がある[30]。

```
hostA% delegated -P7777 CACHEDIR=/var/spool/delegate/cache
hostB% delegated -P8070 MASTER=hostA:7777 SERVER=gopher://gopher.etl.go.jp/
hostC% delegated -P8080 MASTER=hostA:7777 SERVER=http://www.aist.go.jp/
```

経路制御

6章に述べた経路制御は、 MASTERパラメタによって指定する。 MASTERには、中継先のホスト名: ポート番号に加えて、この中継先に中継すべき目的サーバのリストHostListを与えることができる。例えば、

```
MASTER=ripsport:7777:{*.ac.jp,*.go.jp}
MASTER=etlport:7777:*.co.jp
MASTER=ripsport:7777
MASTER=etlport:7777
```

のように指定すると、 *.ac.jpおよび*.go.jpというパターンにマッチするドメイン名を持つ目的サーバへはripsport経由で、 *.co.jpはetlport経由で送られる。いずれにもマッチしないか、あるいは指定されたMASTERへの接続が失敗した場合には、 etlport, ripsport の順で接続が試行される。

マウント機能

5章に述べたように、別のサーバ上にある情報を、あたかもHTTP-delegatedの上にあるように見せる方法である。例えば、 firewall である www.etl.go.jp の上で HTTP 専用delegated を走らせ、以下のようなMOUNT引数を与える。

```
MOUNT="/etl/*       http://etlwww/*"
MOUNT="/gsj/FTP/*  ftp://gsjftp/*"
```

[30] これは主に、文字コード変換・ URLの書き換え・ GopherからHTTPへのプロトコル変換などの処理と、キャッシュの管理を一つのサーバプロセスで行うことによる複雑化を避けるためであった。第1版の終盤の版では、 HTTPクライアントからのHTTPサーバへのアクセス、 GopherクライアントからのGopherサーバへのアクセスに関しては、 Specialistでもキャッシュが可能となっている。

```
MOUNT="/gsj/*        http://gsjwww/*"
MOUNT="/*            http://www.etl.go.jp/*"
```

これにより、例えば firewall 内部にある http://etlwww/welcome.html という情報は、この delegated の利用者からは firewall 上の http://www.etl.go.jp/etl/welcome.html にあるように見える。MOUNT 引数が複数指定された場合には、最初のものから順にマッチングが試みられるので、より特殊なものから一般的なものへという順に引数を与える必要がある。

セキュリティ管理

セキュリティのためのアクセス制御を指定するパラメタとしては、 RELIABLE, REACHABLE, RESTRICT がある。 RELIABLE はこの delegated にアクセス可能なクライアントホスト (またはネットワーク)、 REACHABLE はこの delegated からアクセス可能なサーバホスト (またはネットワーク) を表し、前述の HostList の形式で値を指定する。 RESTRICT はこの delegated が中継可能なプロトコルのリストを指定する。 Fierwall 上で delegated を実行する場合、これらのパラメタは、 DeleGate の使用目的に応じて、以下のように設定することが推奨される。

- 内部クライアントから外部サーバへ
  以下のようにして内部のクライアントホストからのみ受け付け可能とする。

  ```
  RELIABLE= 内部のクライアントホスト（ネットワーク）のリスト
  ```

  この場合、最も単純は方法は、 (1) firewall hostF 上では内部の hostI からのアクセスのみを受理する (2) 内部の hostI は全てのアクセスを受理する、という設定を行うことである[31]。

  ```
  hostF% delegated -P7777 RELIABLE=hostI
  hostI% delegated -P8080 MASTER=hostF:7777 SERVER=http://-/-/
  ```

- 外部クライアントから内部サーバへ
  MOUNT 機能を用いて内部のサーバを外部に公開する場合、中継は内部のサーバへのみ、そのサーバが提供するサービスに関してのみ行い、それ以外を中継しないようにする。例えば、内部にある HTTP サーバを公開する場合には、以下のように指定する。

  ```
  REACHABLE= 内部のサーバホストのリスト
  RESTRICT=!all,http
  ```

  この例の RESTRICT の右辺は、「全てのプロトコルを禁止，ただし http は許可する」ことを指示している。

- 外部クライアントから外部サーバへ
  現在、主に文字コード変換機能の提供を目的として「公開 DeleGate サービス」が全国数か所で行われている[32]。このようなサービスを提供する場合、 DeleGate を介して提供者の内部に侵入が起こらないよう注意する必要がある。そのため、以下のような設定を行う。

  ```
  REACHABLE=*,!{内部のホスト（ネットワーク）のリスト}
  RESTRICT=!all,http,gopher,ftp
  ```

[31]もともと DeleGate はこのような形式での接続とアクセス制御を想定して作られたため、「アクセス制御を信頼して任せるられるホスト」という意味で RELIABLE というパラメタ名が付けられた。

[32]東海インターネットワーク協議会の高木氏 < *takagi@center.nitech.ac.jp*> が公開 DeleGate サーバー覧を <http://www.tokai-ic.or.jp/WWW/delegate/> に作成している。

RFC822・ MIME メッセージのエンコード・デコード

ネットワークニュースおよびメールの転送に用いられる NNTP, SMTP および POP に関しては、クライアント / サーバ間で転送されるメッセージ (RFC822 形式) の MIME コードおよび文字コード変換を行う。まずサーバからクライアントへの中継では、転送されるメッセージヘッダ中の MIME エンコードされた日本語コード (ISO-2022-JP) をデコードする。もし CHARSET パラメタが指定されているなら、メッセージ全体を指定された日本語コードに変換する。逆にクライアントからサーバへの中継では、転送されるメッセージ中の日本語コードを ISO-2022-JP に変換し[33]、もしヘッダ中に ISO-2022-JP が含まれるなら、それを MIME エンコードする。

書き換えによる中継

DeleGate では「中継サーバによる中継」の機能をクライアントやサーバがサポートしていない場合にも、中継が可能である。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、 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 URL を、 (a) から (b) のように書き換える。

（a） `http://hostT:portT/path`
（b） `http://hostD:portD/=@=:hostT:portT/path`

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、例えば

```
GET /=@=:hostT:portT/path
```

という要求を送る。すると 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。目的サーバからの応答中に含まれる URL は、上述のように (a) の形から (b) の形に書き換えられる。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 Proxy-httpd 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[34]。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 トンネル

IP でない接続の上で DeleGate プロトコルを中継するための試みとして、トンネル機能が作られた。トンネルの出口に作られる delegated と、入口にいる delegated の間は固定的に接続され、 DeleGate プロトコルの上で HTTP, FTP, Gopher などのプロトコルが中継される。 TUNNEL パラメタには、入口にいる delegated が出口の delegated を起動して接続するためのプログラムの名前を指定する。

[33] インターネット上では、ニュースあるいはメイルの交換の際に、日本語コードとしては ISO-2022-JP を使用することが合意されている。

[34] 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。

8 DeleGate 第 2 版に向けて

DeleGate 第 1 版での開発・試用・評価を経て、第 2 版以降では以下のような改善、拡張を行うことを考えている。

8.1 接続のキャッシュ・プリフェッチ・固定化

HTTP プロトコルは stateless なプロトコルである。一つの情報の取得ごとに、クライアントからサーバへの接続、クライアントからサーバへの要求送信、サーバからクライアントへの応答送信、接続終了、という手順が行われる。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、その処理自体の比重は高く、実際の情報の送信よりも重いことも少くない。きめの細かいアクセス制御を行おうとするほどこの傾向が高くなる。したがって、もしこの問題を解決することができれば、接続に関する性能とセキュリティの向上の両者に有益である。

- ゲートウェイにおける接続の維持

  中継サーバにより FTP や NNTP などの stateful なサーバと、 HTTP のような stateless のクライアントを中継する場合、 HTTP クライアントからの要求ごとに毎回サーバと接続し直すのは無駄が多い。そこで、一旦遠隔のサーバとの接続を確立したら、その接続を一定時間保持する「connection のキャッシュ」の導入を考える。

- 多段 **DeleGate** 間の固定接続

  第 1 版で試験的に導入した「トンネル」機能を、実用レベルで実現する。 DeleGate 間の継続的接続であるトンネルを「バックボーン」として用い、その上で PPP(Point-to-Point Protocol) [PPP] による通信の多重化を行う[35]。

  例えば、広域ネットワーク上の分散した地点に DeleGate を置き、 DeleGate 間に固定的なネットワークを構築する。各地点でキャッシュを行うとともに、冗長な経路から最適な経路を選択して転送を最適化するというような応用を行う。

- クライアントと **delegated** 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。クライアント / 中継サーバ (delegated) 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、多段 DeleGate 間の固定接続と同じ方式で実現できると考えられる。

8.2 アクセス制御の高度化

DeleGate 第 1 版では、アクセスの制御を一組しか記述できなかった。すなわち、「ホスト A からはホスト B へ、ホスト X からはホスト Y へのアクセスを許す」という記述はできない。このため、対象とする利用者の違いに応じて複数の DeleGate サーバを起動して管理する必要があった。これを解決するために、第 2 版では任意の数の組を指定できるようにしたい。

[35]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。

- ユーザ単位の制御
  コネクション上のユーザ認証機能 [IDENT] による、ユーザ単位の認証とアクセス制御の導入。

- コマンド単位の制御
  プロトコルに含まれる各コマンドの単位でのアクセス制御の記述の導入。

- 汎用アクセス制御エンジン
  既存のプロトコル (NNTP, SOCKS, HTTP, ...) で用いられて来たアクセス制御ファイルの記述をそのまま取り込めるパーサと、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」。

8.3 双方向プロトコルへの拡張

第 1 版の DeleGate プロトコルは、クライアントからサーバに対して情報を与えるだけの単方向のものである。このため、サーバからクライアントに対して情報を与える一般的な手段が無い[36]。第 2 版では、これを双方向のプロトコルに拡張する予定である。これにより、例えば目的サーバに至る DeleGate の連鎖の中途で接続が失敗した場合には別の経路を試みる、経路ごとの負荷状況を収集する、などを実現したい。

8.4 キャッシュ制御

- キャッシュ容量の制御

  第 1 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[37]。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、 DeleGate 第 2 版で実現するべき課題の一つである。

- キャッシュ対象プロトコルの拡大

  第 1 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全般に適用範囲を拡張して、例えば FTP, NNTP などもキャッシュの対象にしたい。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。 NNTP の通信をキャッシュする機能によって、一つのサーバを DeleGate 経由で共有する一方、一つの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。

- 自動更新とミラー

  ある情報のキャッシュの更新は、ユーザがその情報にアクセスした時に行われる。この方法には、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より前に自動的に更新をかけるなどの対策が必要である。ただしこの機能は、 DeleGate の外部の機能として実現すべきかもしれない (httpd など他の caching proxy でも共通に利用可能なものとして)。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サービスする、いわゆる「ミラー」サーバの機能を実現する。

[36] HTTP では、応答が要求同様 MIME 形式であるため、応答メッセージのヘッダの中に DeleGate の情報を埋め込む方法を、第 1 版では実験的に行った。

[37] 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。

8.5 複数 delegated の統括管理

複数の delegated 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 delegated の起動と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。

8.6 その他

以上の他にも、今後の DeleGate で導入、実験を試みたい多くの課題がある。

- マウント機能の適用範囲の拡大: FTP や Gopher プロトコルへの適用、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なマウント機能。サーバ毎に MOUNT パラメタで変換規則を指定する必要の無い「オートマウント」機能。
- **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。
- 経路制御のためのプロトコル: DeleGate 間ネットワークでの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 [BGP] の利用も検討する。
- 輻輳制御: 経路の転送遅延状況、ユーザごとのプライオリティなどに基づく転送の制御。
- 文字コード変換の適用範囲の拡大: telnet や X プロトコルの中継時の文字コード変換。
- **stateful CGI**: stateful な protocol のための CGI(Common Gateway Interface) の開発。

9 おわりに

開発中のプロトコル中継システム DeleGate について、その開発の背景と現状、実現方式の概要を述べた。 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プロトコルの中継という段階を導入することにより、様々な機能が、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導入でき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り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経て、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と思われる。

DeleGate という名前は、オブジェクト指向システムのクラス階層の実現手法である *delegation* に由来している。 WWW は世界中に分散したハイパーテキストシステムを実現したが、世界中に分散したオブジェクト指向システムに到達するには、やはりクラス階層の導入が必要だと考えられ、 DeleGate は経路制御機構などによりその端緒をひらくことを目指している。

今後も、実用システムとしての完成度を高めると同時に、新たな分散情報システムの方式への手掛かりを得る探針として活用して行きたい。

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、工技院計算情報センター (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上で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、数十人の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。以上の方々に感謝いたします。

参考文献

[HTTP] HTTP: Hyper Text Transfer Protocol,
<http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html>

[GOPHER] 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>

[URL] T. Berners-Lee: Uniform Resource Locators (URL), A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network, Inernet Draft, 1994.
<ftp://ds.internic.net/internet-drafts/draft-ietf-uri-url-08.txt>

[PROXY] A. Luotonen and K. Altis: World-Wide Web Proxies, WWW'94 Conference, 1994.
<http://info.cern.ch/hypertext/WWW/Proxies/>

[MIME] 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>

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

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

[SOCKS] D. Koblas and Y-D. Lee: <ftp://ftp.nec.com/pub/security/socks.cstc>

[IDENT] M. St. Johns: Identification Protocol, RFC1413, 1993.
<ftp://ds.internic.net/rfc/rfc1413.txt>

[ISO-2022-JP] J. Murai, M. Crispin and E. van der Poel: Japanese Character Encoding for Internet Messages, RFC1468, 1993. <ftp://ds.internic.net/rfc/rfc1468.txt>

[L10N] 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 Vol.1, No.2, 1994. <http://www.ntt.jp/people/takada/docs/www-j10n/>

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

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

Appendix-1: DeleGate の入手・作成・利用法

入手方法

- **Anonymous-FTP**: DeleGate の最新版は、 anonymous-FTP により以下のディレクトリから配布しています。

ホスト:	etlport.etl.go.jp [192.31.197.99]
ディレクトリ:	pub/DeleGate
ファイル名:	delegate*.tar.gz

- フリーソフトウェア集 **CD-ROM**: 最新版ではありませんが、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ます。

DeleGate1.5.9	インターネットマガジン, 創刊号付録 CD-ROM (94/09)
DeleGate1.6.4	日本 DEC, DEC ALPHA 用フリーソフト '94 CD-ROM (94/09)
DeleGate1.6.5	日本 SUN ユーザーズグループ (NSUG), '94 CD-ROM (94/09)
DeleGate1.7.5	日本 UNIX ユーザ会 (JUS), '94 CD-ROM (94/10)
DeleGate1.9.8	インターネットマガジン, Vol.2 付録 CD-ROM (94/11)

作成方法

現在のところ DeleGate は Unix 上でのみ動作します。以下では、あなたが入手した DeleGate の版を *VER* と表します。

(1)	% gzip -d < delegate*VER*.tar.gz | tar xfv -
(2)	% cd delegate*VER*/src
(3)	Makefile の編集 システムの種別やローカルホストの環境に合わせて変数を設定して下さい (*)
(4)	% make delegated および delegatedR (ネームサーバ対応版) が作られます。
(5)	% make delegatedS delegatedS (SOCKS 対応版) が作られます。

(*)BSD 系の Unix や OSF/1 では標準の設定のままで良いはずです。 SysV 系の Unix では、配布ファイル中の DELEGATE_CONF.sys を参考にして src/Makefile 中の変数の設定を変更して下さい (必要な変更は 2 ~ 3 行程度です)。

起動方法

DeleGate サーバの起動の方法については、配布ファイル中の README を読んで下さい。パソコン上のクライアントから利用する場合には、配布ファイル中の HowToMacMosaic.jp(電総研 山名氏提供), HowToWinMosaic.jp(東大理学部 白橋氏提供) を参考にして下さい。不明な点は delegate@etl.go.jp まで問い合わせて下さい。以下に簡単な例を示します。

　例： JIS コードへの変換付き HTTP 用 DeleGate サーバを hostA で起動し、これを hostB の上で Proxy_http 対応クライアント (mosaic) から利用する。

```
hostA% delegated -P10080 SERVER=http://www.ntt.jp CHARSET=JIS
hostB% setenv http_proxy http://hostA:10080/
hostB% mosaic
```

開発・改良への参加

- DeleGate メーリングリストにて、コメントをお待ちしています。

アドレス:	delegate@etl.go.jp
開設:	94 年 7 月 21 日
参加者数:	185 名 / 115 組織 (94 年 11 月 13 日現在)
記事流量:	平均 40 通 / 週
バックナンバー:	etlport.etl.go.jp:pub/DeleGate/ML/

- 参加方法: DeleGate メーリングリストに参加を希望される方は、以下のようなメールを送って下さい。自動登録されます。

```
To: services@etl.go.jp
Subject: listserv subscribe delegate   Makiko Tanaka
                                       (貴方の名前と姓)
(本文は不要です)
```

Appendix-2: Anonymous-FTP による DeleGate の配布状況

DeleGate 配布ファイル (delegate1.*.tar.gz) に対する anonymous ftp による転送の統計 (6/12 配布開始 ~ 11/3 現在)

転送回数:	4501	
取得者数:	1520	(重複あり *1)
ホスト数:	929	
組織数:	316	
ac.jp:	150	教育・研究機関
ad.jp:	9	ネットワーク運営組織
co.jp:	94	企業
go.jp:	16	国公立機関
or.jp:	19	非営利団体
jp:	3	その他
edu:	3	米国教育・研究機関
com:	18	米国企業
de:	2	ドイツ
fr:	1	フランス
uk:	1	イギリス
(不明)		(*2)
クラス B:	11	
クラス C:	6	

(*1) この数は、 anonymous-FTP サーバ利用時にユーザが入力する自分の電子メイルアドレスを元にしているので、タイプミスなどによる重複カウントを含んでいる。
(*2)IP アドレスからドメイン名を逆引き検索できなかったために、ドメイン名が不明のもの。判明分との重複関係は不明。