第 15 部

パソコン通信との相互接続実験

第 1 章

はじめに

1.1 パソコン通信との相互接続実験

WPNC ワーキンググループは、一般に「パソコン通信」と呼ばれているホストを主体とした BBS サービスを WIDE インターネットに接続に当たって発生する諸問題を解決するための活動を行なっている。

ワーキンググループの活動は 1992 年度から行なっており、1992 年度は主に電子メイルの相互交換について研究を行ない、本 1993 年度は電子メイルの発展とサービスを提供するホストへのアクセスについて研究開発を行なって来た。

BBS サービスの中では草の根 BBS と呼ばれる非営利のサービスもあるが、今回共同で研究を行なっている相手は商用 BBS であり、このためホスト接続のための技術的課題ばかりでなく、電気通信事業などに関連する社会的課題の解決も必要となる。

まず、パソコン通信をつなぐ積極的な動機としては、次のような項目があげられる。

- 国内においては、インターネットに比べてパソコン通信の方が広く一般に認知されている。また、特に大きな商用 BBS においては全国展開がなされているため、何らかの形でつなぐことてインターネット全体としての裾野の広がりが期待できる。

- 商用 BBS を利用した様々な研究活動が実際にいくつかの分野で展開されており、そのような活動をインターネットと接続することは双方にとって研究を進める上で非常に効率を高める。

- 商用 BBS の国際的な関係は様々な制限から非常に限られているが、インターネット側は当初より国際的な協調関係の中で拡大してきたものである。したがって、商用 BBS がインターネットと接続されることで、商用 BBS の対外的な関係を進めることが期待される。

- 商用 BBS 上にはデータベース・サービスなど様々なサービスが提供されており、それらのサービスがインターネットから利用できるようになることは、インターネット上での種々の研究活動を進める上で、大きなメリットとなる。

一方、パソコン通信をつなぐにあたって、慎重に考慮すべき点には、以下のような点がある。

- 国内のインターネットと商用 BBS は本質的に文化的、社会的および技術的背景が異なるため、これら両方の世界を接続するに当たっては、十分に慎重な擦り合わせが必要となる。例えば、日本語の利用や、漢字コードの変換の問題、あるいは匿名性に関する意識の違いなど、潜在的に大きな摩擦を発生させる可能性がある。

- 国内のインターネットにおいてはこれまで商用サービスの利用に関して、非常に消極的でコンセンサスも得られておらず、場合によっては拒否反応を招きかねないため、全体のコンセンサスを徐々に形成する必要がある。

- 商用 BBS には複数のサービス提供者が存在しており、これらのサービス提供者間で WIDE インターネットとの接続により大きな不公平が生じないように注意する必要がある。

- 商用 BBS 間の相互接続は事業に伴う様々な制限が存在するため、インターネットを介して商用 BBS 群が相互に接続するような状況は好ましくない。このため、制限を破らないような技術が必要になる。

以上の点を踏まえて、WPNC では WIDE インターネットと商用 BBS との接続に当たって、次のような方針に沿って実験を進めた。

- 影響の評価や、必要な技術のデバックを行うために、実験を次の 3 フェーズに分け徐々に実験範囲を拡大する。

  フェーズ I WIDE バックボーン内での実験

  フェーズ II WIDE インターネット内での実験

  フェーズ III 国内インターネット全体での実験

- 電子メイルの交換ついてはドメイン名により制限を行う。

- メイル交換においては、それぞれの側のルールにあわせることを原則とし、インターネット側に送られて来る電子メイルは RFC に則った形式となるように調整する。

- 必要な技術の開発を行い、それらを広く一般公開する。

- 実験のアナウンスに関しては責任がとれるような形で、可能な限り広い範囲に広報する。

また、これらの方針については、商用サービスとの初めての接続ということでもあり、関係するネットワーク・プロジェクトや日本ネットワークインフォメーションセンター (JNIC[1]) との連絡を密に行った。さらに、国内のインターネットのネットワークについて技術的な検討を行う JEPG/IP(Japan Engineering Planning Group for IP) において、接続のためのフィルタリングの方法などについて議論を行った上で、全体的な調整と連係の下で実験を進め、また実験の経過に付いても逐次報告を行っていった。

[1] 現在は JPNIC へ組織の改編が行われた

この実験の結果を受けて、JEPG/IP では商用など運用方針が異なるネットワークとのメイルの相互接続に当たっては、メイルそのものをフィルタリングせずに、通信を望まない側が IP レベルでのフィルタリングを行うことが望ましく、それを実現するために接続する側がフィルタリングが可能になるようにアドレスを区別することが望ましいとの勧告を行っている。

1.2　ワーキングループの活動記録

WPNC ワーキンググループは 1993 年度には以下のような活動を行った。

1. 電子メイル相互接続の組織 (ドメイン) の拡大
2. 電子メイルにおける日本語の取扱の問題の規格に基づく調整
3. BBS ホストへのインターネット経由での telnet による端末アクセスの実現
4. 接続に伴う新たな技術の開発
   (a) 電子メイルにおける MIME ヘッダの導入と啓蒙
   (b) ファイル転送プロトコルを含む telnet の開発と実験 (telnetx と呼ぶ)

また 1993 年度の活動の概要の履歴を表 1.1に示す。

表 1.1: 1993 年度の活動と関連する事象

1993 年	5 月	13 日 回線速度増強 (2400bps → 9600bps, NIFTY-Serve)
	5 月	17 日 電子メイル交換範囲を国内インターネット全域に拡大 (NIFTY-Serve, PC-VAN)
		接続形態の変更 (PC-VAN)
	6 月	3 日 RFC における ISO-2022-JP の定義 (RFC1468)
	6 月	17 日 MIME ヘッダの導入実験の開始 (NIFTY-Serve)
	7 月	1 日 MIME ヘッダによる運用実験の開始 (NIFTY-Serve)
	8 月	1 日 `Reply-To:` への対応 (NIFTY-Serve)
	9 月	9 日 telnetx αバージョンの WIDE 内リリース
	10 月	30 日 接続形態の変更 (NIFTY-Serve)
1994 年	2 月	1 日 telnet サービスの開始 (NIFTY-Serve)

これらの実験のうち、MIME ヘッダおよび telnetx の技術的な内容については次章以降で説明を行う。

1993 年度の相互実験においては、当初に計画した目標を実現できたと評価できる。しかし、一方でインターネットあるいは各 BBS の中で、ユーザへの広報あるいは啓蒙活動

に不十分な点があり、一般的な広報活動の困難さおよびそのためのメディアや技術の整備の不完全さが課題となった。

今後の活動予定は、1993 年度の実験を引続き行なうとともに、課題の解決を含めて社会的な側面や、商用サービスとインターネットの関わり等にも目を向けて行く予定である。

第 2 章

パソコン通信ネットとのメール交換システムと、MIME ヘッダの導入

2.1 はじめに

UUCP 接続のネットワークではじめられた電子文書の交換は、現在の TCP/IP 接続ベースのネットワークにおいても頻繁に行われている。日本では、全角文字を含む拡張文字集合を用いてこれらの電子文書を作成し、交換している。特に電子メールと電子ニュースで交換される電子メッセージについて国内では、『JUNET の手引き』[202] をはじめとするとりきめが交され、メッセージのヘッダ部においては「NVT-ASCII 文字集合」と呼ばれるアスキー文字集合のサブセットにあたる文字集合のみを、メッセージのボディ部においては「JUNET コード [196]」と呼ばれる ISO-2022 エンコーディング方式によって実現される文字集合を使うように指導されている。しかし、WIDE project の WPNC ワーキンググループが中心になってすすめていたパソコン通信ネット—インターネット間の電子メールシステムの相互接続実験を通して、メッセージのヘッダ部での日本語を表現できる文字集合の使用を求める声が高まってきた。

WPNC ワーキンググループによって実験が開始された直後に、電子メールや、電子ニュースの電子メッセージのヘッダ部において ASCII 文字集合以外の文字集合の使用を可能にする符号化規格を取り入れたメッセージ記述のフォーマット、「MIME」が、RFC となった。

2.2 パソコン通信との相互接続

WIDE project の WPNC-WG では、昨年度よりパソコン通信と WIDE インターネットとの間で電子メールシステムの相互接続実験を行っている。この実験の初期段階でパソコン通信の社会ではインターネット社会での「常識」がほとんど通じないことを思い知らされることとなった。特に日本国内の電子文書の交換における文字コードの重要性と、ヘッダについてのみとはいえ、電子メッセージに使用する文字集合に制限を加えているのは不自然であることを痛感することとなった。これらの問題は尽きることがなかったが、電子メールのヘッダ部については早急な対処を迫られることとなった。

通常、パソコン通信において、ユーザ ID が割り当てられているが、この ID 自体は、意味をもたない英数字を使ったものである。この ID をメールアドレスに使用することは

決めたもの、ユーザ個人の判別がむずかしいという理由で、パソコン通信で実名同然に使われているハンドル名と呼ばれる Shift JIS コードの文字列データを付加することになった。

この実験では、将来的にはインターネット全域に電子メールを送り出すことを目的としているために、RFC822[194] 準拠であることが要求された。ハンドル名や、日本語 Subject は、Shift JIS コードによる文字列であったので、拡張ヘッダフィールド "X-From:" を用意し、このフィールドに JUNET コードに変換したハンドル名を埋めこんだ。

しかし、実際に運用してみると、JUNET コードに含まれる ISO-2022 のコード切り換えシーケンスを理解しない MTA が数多く存在し、送信エラーが発生して送信できないことが続発した。また、ネット内のメールシステムのインターフェイスでは、日本語 Subject が使用が許されているために、ユーザへの呼びかけにも関わらず、Subject フィールドにも JUNET コードの文字列が使用されてしまうこともあった。

このような問題の解決策として、NIFTY-Serve からのメールに対して当時 RFC になりたての MIME (Multipurpose Internet Mail Extensions) を採用することになった。

現在 NIFTY-Serve からインターネット内のサイトに送信されるメールは、JUNET コードに変換した後で『旧 MIME part two』の仕様でエンコードした日本語ハンドル名を From: フィールドに送信者名として使用している。日本語の Subject も同様にエンコードして、Subject: フィールドに使用している。もちろん、インターネット側から送られてくる MIME メッセージは、デコードとコード変換をした後に、NIFTY-Serve 内に送られることになっている。

2.3 MIME とは

MIME(Multipurpose Internet Mail Extentions) は、電子メール、電子ニュースで交換されるメッセージのフォーマットを定義した RFC822 に対する拡張定義である。この既約は、ボディ部への適用を定義した『MIME part one[174]』と、メールのヘッダ部への適用を定義した『MIME part two[176]』の 2 つの部分に分けられる。

2.3.1 MIME part one

『MIME part one』では、メッセージのボディ部に種類の異なる複数のデータを取り入れるのに必要な宣言の仕方を定義している。この宣言により得られる情報をもとに、送られてきたデータを再生することができる仕掛けになっている (図 2.1参照)。

現在使用できるデータは、 text、audio、image、video、message の 5 種類である。MIME をエンコーディングを適用すると、このよう種類の異なるマルチメディアのデータを混在させて送受信することができる。つまり、異機種間であっても簡単にマルチメディア・メッセージを交換できるのである。

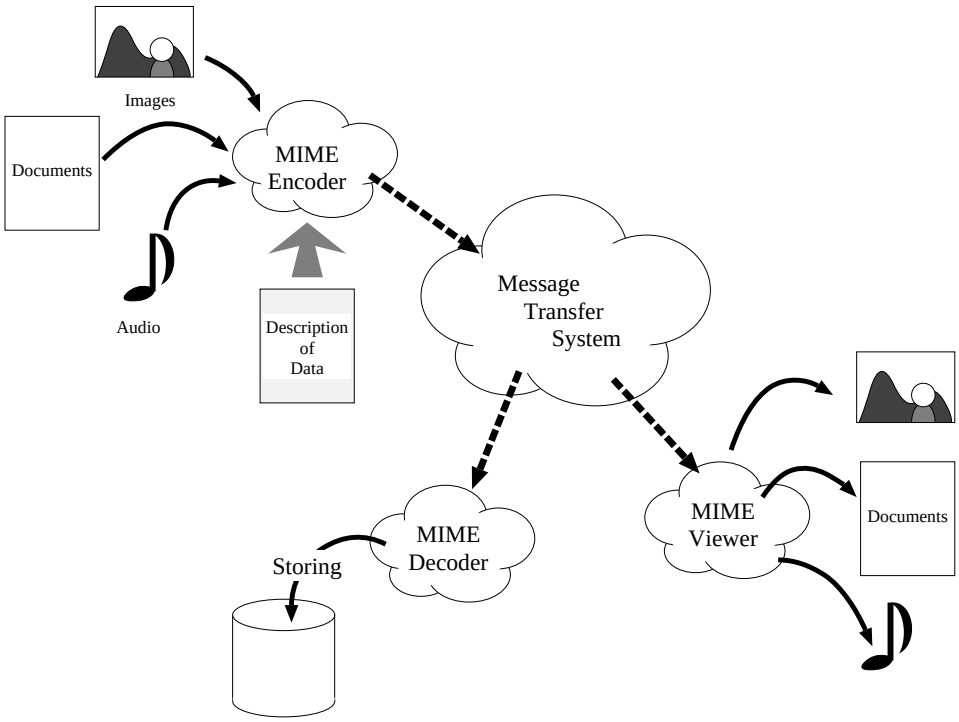

図 2.1: MIME エンコーディング処理システム

2.3.2　MIME part two

一方、『MIME part two』では、メッセージのヘッダ部における非 ASCII 文字の使用法を定義している。

元来、(通常 MTA と呼ぶ) メッセージの配送を受けもつアプリケーションの混乱を避けるために、メッセージのヘッダ部には ASCII 文字集合から、メールシステムの制御に使用する文字 (メタキャラクタと呼ぶ) を除いた NVT ASCII という文字集合のみの使用が許されている。

RFC822 に従うとメタキャラクタを通常の文字として使用するためには、クウォーティング (quoting) を適用しなくてはならない。しかしクウォーティングを理解しない MTA が配送システムにあると混乱が生じることになる。(調査によるとクウォーティングを理解しない MTA の方が多いくらいであるそうである。)

このような配送システムを混乱させないようにメッセージのヘッダをエンコーディングし、配送後に復元させるために考案されたエンコーディング手法を定義しているのが『MIME part two』である。

2.3.3　MIME エンコーディング

MIME エンコーディングでは、配送の途中でシステムを混乱させないようにするために、すべてのデータは MTA が制御文字に使わない文字のみで表現されるデータへと変換

=?(文字集合名)?(エンコーディグ方式名)?(エンコードされたデータ)?=

図 2.2: エンコード文字列の宣言

する。この変換において適用する変換方式として、現在 “Q–Encoding” と “B–Encoding” の 2 つの方式が考案され利用されている。

Q エンコーディングでは、ASCII 文字以外の文字を 16 進コードで表現する。また、例外としてスペース文字は、“_(アンダースコア)” で表現する。Q エンコーディングが適用できるデータは、データのほとんどが ASCII 文字集合に含まれているが、一部特殊な文字も出現するようなテキストのみである。主にアクセント記号などを使用するラテン語圏の文字集合で記述されたテキスト等に適用される。このエンコーディングの特徴は、デコードしなくてもだいたいの意味を理解できることである。

B エンコーディングは、BASE64 というアルゴリズムを使用したエンコーディング方式である。このエンコーディング方式は、従来の uuencode の代りに使われることを想定している。エンコードの効率は uuencode と同じであるが、エンコード後のデータに配送システムにおけるメタキャラクタを含まないことが特徴である。

2.4 MIME part two と日本語メッセージ

“MIME part two” を適用すれば、ASCII コード以外の文字集合をメッセージヘッダで使用することができる。これが多言語ヘッダと呼ばれるものである。

ヘッダで使用できる文字集合は、あらかじめ登録されたもののみである。現在日本語の記述に使用できる文字集合は、いわゆる JUNET コードが “iso-2022-jp” として登録されている。この文字集合は、メッセージのボディ部と共通な概念であるが、ヘッダ部とボディ部の文字集合はそれぞれ独立したものとして扱われる。

ヘッダ部で使用された非 ASCII 文字列を MIME エンコードした場合の宣言フォーマットは、図 2.2のようになっている。

最初のフィールドである文字集合名には、例え “us-ascii” や, “ISO-2022-JP” のような文字列が使われる。このフィールドでは大小文字を無視するので、“iSo-2022-Jp” も同じ文字集合を指すことになる。エンコーディング方式名には、データがエンコードされているエンコード方式として Q または B のいずれかを指定する。日本語を適用する場合には、B–encoding を使用することが推奨されている。

実際にエンコーディングしたメッセージヘッダの例を図 2.3に示す。

このように一見暗号のように見える MIME エンコード後のメッセージヘッダであるが、拡張されたクォーティングが適用されているわけである。

(エンコード前)
From: 藤本真吾 <shingo@wide.ad.jp>
Subject: 日本語ヘッダのテスト

(エンコード後)
From: =?ISO-2022-JP?B?GyRCRiNLXD8/OGMbKEI?= <shingo@cs.uec.ac.jp>
Subject: =?ISO-2022-JP?B?GyRCRnxLXDhsJVglQyVAJE4lRiU5JUgbKEI=?=

図 2.3: 日本語ヘッダのエンコード

2.5　MIME part two の利用

MIME の利点は、メーラなどの個人レベルでのアプリケーションによる対応ができる点にある。MIME エンコードされたメッセージを作成／表示するには、MIME をサポートしたソフトウェアを利用すればよい。『MIME part two』対応のソフトウェアの開発は、主に日本国内で行なわれており、毎週のように国内向けネット・ニュースで発表されている。

ここでは 1993 年 3 月現在公開されている『MIME part two』をサポートしているソフトウェアを紹介する。著者が利用したことがあるものに限られているので、抜け落ちているものについてはご容赦願いたい。

一般にエンコーディングシステムと、デコードディング (表示) システムは、独立したものになっているので、それぞれに分けて紹介することにする。つまり、自分では MIME エンコーディングをしないというユーザであれば。デコーディングシステムのみをインストールすればよい。

ここに挙げたソフトウェアは、anonymous ftp サイト ftp-pdl.cs.uec.ac.jp にアクセスして入手が可能であるが、 archie などのサービスを利用してもよりのサイトから入手することが望ましい。

2.5.1　デコーディングシステム

様々なインターフェイスが用意されており、エンコードシステムに較べるとかなり充実していてユーザの好みにあわせて選択できる。

- **metamail**
  コマンド metamail は、MIME メッセージのインタプリタである。しかし、同時に配布されるエンコードプログラムや、再生システムが多くのアプリケーションから呼び出されて利用されることのほうが多い。MIME メッセージのボディ部、ヘッダ部のデコードと再生が可能。

- **metamail.el**
  先に紹介したコマンド metamail の Emacs 用インターフェイスプログラムである。後で紹介する MIME 対応のニュースリーダ gnus は、これを内部から利用している。

  キット： /pub/MIME/src/for-emacs/metamail.el.Z

- **MH version 6.8.x (jp2)**
  MH version 6.8 は、option MIME つきでコンパイルすると、ボディ部に含まれる MIME メッセージの再生が可能になる。jp2 版では、さらにフォーマットファイルから呼び出し可能なデコード用の関数と、ヘッダの MIME エンコーディング機能が拡張されており、多言語ヘッダのサポートがされている。

  キット：
  /pub/MIME/src/mh/mh-6.8.1-to-JP2c-patch.gz
  (MH-6.8.1 用パッチ) 参考：
  /pub/MIME/doc/setting/mh-jheader-kit.tar.gz

- **VM**
  MIME サポート用のパッチが配布されており、metamail をフィルタとして起動してサポートする。

  キット：
  /pub/MIME/src/vm-5.32-mime.tar.Z

- **Elm**
  オリジナルのパッケージから MIME がサポートされている。コマンド metamail を内部から呼び出して MIME メッセージの再生を行う。

- **tinymime.el**
  ヘッダ部のみのデコードをサポート。GNU Emacs の elisp だけで作成されている独立プログラムであることが特徴。MIME part two(多言語ヘッダ) をサポートする最小限の機能が盛りこまれている。

  キット： /pub/MIME/src/for-emacs/tiny-mime-3.2.tar.Z

- **FromMime**
  ヘッダ部のデコードのみをサポート。こちらは、C 言語で書かれたプログラムで、通常フィルタとして利用される。後述する ToMime とセットで配布されている。

  キット： /pub/MIME/src/mimehead.tar.gz

2.5.2　エンコーディングシステム

日本語ヘッダを MIME エンコーディングするソフトウェアの実装については、その仕様について検討の余地があり、『MIME part two』は今年の 9 月に更新されたばかりであることもあって、最新の『MIME part two』の仕様に準拠したものは残念ながら少ない。

- **ToMime**
  RFC1522 の仕様をサポートしている。FromMime 同様フィルタとして利用できるので、多くのメーラでの利用が可能となっている。

- MH version 6.8.x (jp2)
  更新前の RFC1342[203] の仕様をサポートしている。もともとは、MH で日本語を扱うためのパッチキットであるが、MH の日本語化と同時に多言語ヘッダを扱う拡張機能が追加される。ボディ部も含めた MIME メッセージのエンコード、デコードを外部プログラムを呼び出さずにできることが特徴。

- **mime.el**
  更新前の RFC1342 の仕様をサポートしている。汎用エディタ GNU Emacs から後述する metamail パッケージ中のエンコードプログラムを起動するフロントエンド・プロセッサ。ボディ部の MIME エンコード、デコードも可能。

  キット:
  /pub/MIME/src/for-emacs/emacs-mime-tools.shar

2.6　問題点と今後の課題

MIME part two は、RFC822 で定義されているクウォーティング機能をサポートしていない MTA を介しても正しく配送される。しかし、エンコードされるときにデータの冗長度が増すため、結果として 1 つのフィールド長が長くなる。 1 行の長さが 75 文字を越える場合には、フォールディング (folding) とよばれる改行操作が行われるが、このフォールディング機能をサポートしていないアプリケーションが多数存在することがわかった。フォールディングは、RFC822 で要求されるのもっとも基本的な機能の 1 つであるが、 MIME part two が使用されるまで見落とされていたものと考えられる。このようなアプリケーションは、新しいバージョンでは改善されていることが多いので、早急な対応が必要である。

また、MIME part two の仕様が現在のインターネット・ドラフトの版になるまでに大きく変更されてきたので、一般ユーザへの普及にまでは至っていない。各種サポートツールのおかげでユーザ・インターフェイスが簡略化されてきたとはいえ、MIME を利用しているユーザの数はごくわずかである。今後は、日本国内での電子メッセージの標準フォーマットとしての普及とユーザレベルでの対応が課題である。

第 3 章

telnet によるパソコン通信との相互接続実験

3.1 概要

WIDE/WPNC ワーキンググループでは、インターネットと一般に「パソコン通信」といわれている集中型センター形式の電子掲示版、情報提供サービス組織との接続実験を行っている。現在、第一段階の目標である電子メールによる相互接続が完了し実際に運用されている。そして次の段階の接続実験としてインターネットからパソコン通信を利用可能とする為、telnet による接続実験を行なう。

telnet は複雑な接続プロトコルを持ち、また実装されている機能に応じて様々なバージョンが存在している。これはアーカイブサービスをサポートする上でいくつかの問題を引き起こしている。本研究では「パソコン通信」との接続、及びそのアーカイブサービスを受ける為に telnet に改良を加えた telnetx コマンドを設計・実装した。本論文ではプログラムの実装と運用に当たっての問題点と、その解決方法について述べる。

3.2 接続形態

当初 telnet による接続実験は WIDE と NIFTY Serve との間でのみ行っていた。現在 Nifty Serve は IIJ に専用線 (192K) で接続されており、電子メールもこの回線を利用している。NIFTY Serve 側のホストは、

r2.niftyserve.or.jp

である。これは telnet プロトコル接続をサポートする専用マシンで、NIFTY Serve との ROAD2 接続をサポートする。最初これらのネットワークに対するルーティング情報自体はインターネットに流れてきているが、IIJ 側でアクセス制御を行い 133.4(WIDE バックボーン) と 133.160(for maintainance) からのアクセスのみを許していた。しかし現在ではすでに運用状態に入っておりインターネット全体に広く公開されている。

WIDE の計算機 sh.wide.ad.jp からのアクセスの様子を以下に示す。

```
tani@sh[1]$ telnet 192.47.24.5
Trying 192.47.24.5 ...
Connected to 192.47.24.5.
Escape character is '^]'.
```

```
Enter Connection-ID  --->SVC
Enter User-ID  --->KFC01704
Enter Password ---> ....

ようこそ NIFTY-Serve へ
 Copyright (C) 1993
by NIFTY Corporation
....
```

3.3　telnet 接続における問題点

telnet による接続を通常の電話回線を用いた接続と比較した場合問題となるのが、パソコン通信側のアーカイブサービスの利用方法である。インターネットでは anonymous ftp を用いたアーカイブサービスが一般的だが、パソコン通信でこれを直接利用する事は難しい。

一方多くのパソコン通信では BBS 内で B-plus や ZMODEM といったファイル転送プロトコルを用いてアーカイブサービスを受けることができる。パソコン等で電話回線を利用する多くの通信ソフトウェアは、これを用いたファイルのアップロード・ダウンロード機能を用意しているので問題ないが、telnet にはそのような機能は用意されていない。

また国内のパソコン通信で使われる漢字コードは当然 8bit の Shift JIS(一部 EUC も利用可能なところもある) が中心である。古い telnet ではプロトコル的に 8bit コードを通さなかったり、実装上の問題から一部の 8bit コードが通らなかったりする。

これらの問題を解決する為に、本論文では一般に入手可能な telnet のソースコードを元に、 B-plus や ZMODEM、kermit 等の転送プログラムを利用可能にした。これを telnetx と呼ぶ事にする。

3.4　telnet について

telnetx について紹介する前に telnet 自身の主なバージョンとその機能について紹介する。telnet プロトコルは何度も RFC が更新されており、そのたびに機能拡張が施されている。そのため商用 UNIX システムでこれを完全に満たした telnet を用意しているものは少ない。多くの場合 4.3BSD や 4.3BSD-tahoe のバージョンである。以下に主だった telnet のバージョンと機能を紹介する。[204]。

4.2BSD の telnet

これはもっとも最初の頃の telnet で、古い NEWS-OS や SystemV+4.2BSD ベースのワークステーションなどで利用されていたバージョンである。当然バグも多いようだ。

4.3BSD の telnet

多くの 4.3BSD ベースのマシンベースのマシンはこれがベースとなっているようだ。
例) SunOS NEWS-OS UNIOS-B

4.3BSD-tahoe の telnet

- BINARY オプションがサポートされ 8bit データが送れるようになった。
- それまで 1 つだったソースコードが複数に分割されている。
- UNIOS-Mach 1.20 の telnet がこのバージョン。

March 1, 1990

- LINEMODE オプションのサポート。
- TERMINAL-TYPE(利用可能なターミナル名をサーバに列挙する機能) オプションのサポート。
- NAWS オプションをサポートしてウィンドウサイズを伝えることが可能。

June 20, 1990

- ENVIRON オプション、XDISPLOC オプションを追加し環境変数 DISPLAY などを伝えることができる [205]。
- LINEMODE における SOFT_TAB, LIT_ECHO モードの追加。
- -l user の追加。
- -e escape char の追加。

Februrary 22, 1991

- NET/2 のソースコードをベースとしている。
- Kerberos を使った認証機能 (AUTHENTICATE) のサポート [206][207]。
- 暗号化 (ENCRYPT) をサポート。
- libtelnet を用意。

4.4BSD バージョン

- NET/2 と README の内容は同じ。
- ソースコードの中身はかなり変更されている。
- ソースコードから ENCRYPT 関係が削除されている。
- ENCRYPT + AUTHENTICATE → AUTHENTICATION に変更。

January 19, 1994

- telnet の最新バージョン
- 4.4BSD のコードを反映させている
- 多くのオペレーティングシステムに対応
- 新たに Svr4, Solaris 2.X, HP-UX 等に対応

3.5　実装

今回の実装に当たって 2 種類の方法を検討した。

- 転送プロトコル内蔵方式
- 外部プログラム起動方式

3.5.1　転送プログラム内蔵方式

前者は telnet の内部コマンドとして B-plus や ZMODEM による転送プログラムを内蔵してしまう方法である (図 3.1)。

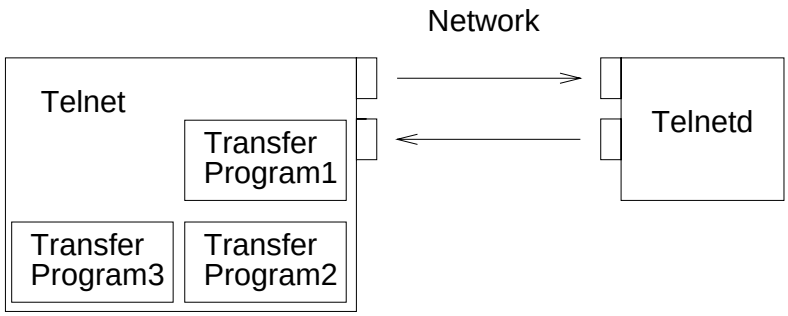

図 3.1: 転送プロトコル内蔵方式

この方法は telnet の内部構造をそのまま活かした実装が可能だが、新しいプロトコルが出るたびに telnetx の変更が必要になりあまり効率的ではない。また新しいプロトコルをサポートするたびにプログラムのサイズが大きくなるのも問題である。

3.5.2 外部プログラム起動方式

後者は転送プログラム自体は telnet の外に置いている。 telnetx は本来ユーザの端末に接続されている入出力を新たに作成した仮想端末 (pty) につなぎかえる。外部プログラムを実行中 telnetx は全てのコードを透過的に転送するのである (図 3.2)。

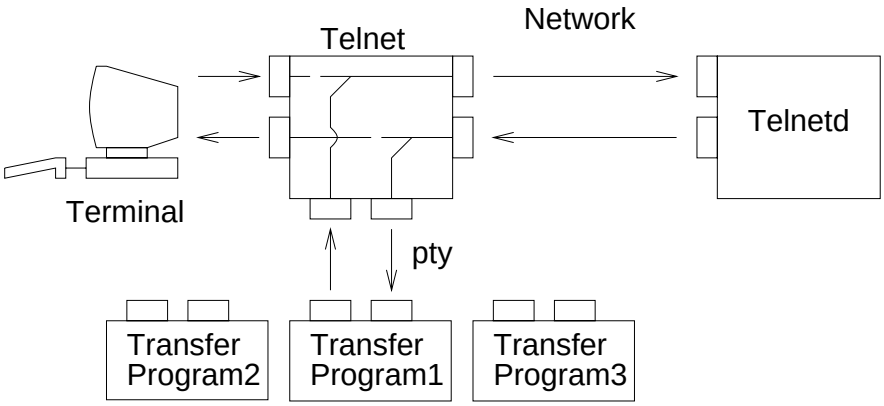

図 3.2: 外部プログラム起動方式

この方式ではサポートする転送プログラムが増えても、 telnet 側の変更が最小限で済む事や転送プログラムの単独のデバックが可能な点、kermit など既存の UNIX 上のファイル転送プログラムも起動する事ができるなど有利な点が多い。また多くの転送プログラムは独自の端末コントロールを行っており、仮想端末を使うことでプログラムの変更も最小限で済む。そこで本研究では外部プログラム起動方式を採用した。

3.5.3 ftp との違い

telnetx と ftp の違いだが、ftp はそれ自体がファイル転送のためのプロトコルであり、TCP/IP の専用ポート番号をもったアプリケーションとなっている。これに対し telnetx はファイルの転送作業そのものは外部プログラムに全て任せている。telnetx からみた場合通常の端末からの入出力をやっているのと何ら変わる所はないわけである (図 3.3)。

唯一異なるのが本来の telnet では特別な処理を行うためのエスケープ処理で、telnetx では外部プログラムが起動している間はエスケープ処理を行わないようにしている。

3.6 プログラムの説明

プログラムは、

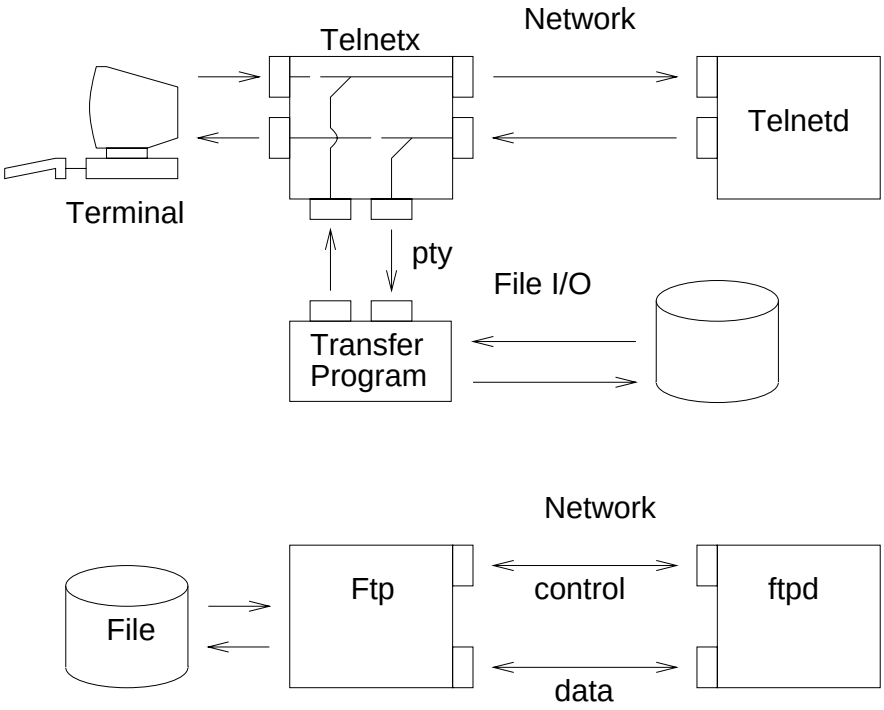

図 3.3: ftp と telnetx の違い

- telnetx 本体
- 転送プログラム
  - B-plus: bp
  - ZMODEM: sz, rz...
  - kermit: kermit
  - transit: transit

から構成される。開発は主に BSD/386 で行ったが、現在

- BSD/386
- SunOS4.1.X
- UniOS-B, UniOS-Mach
- linux

で動作を確認している。

telnet は当初 BSD/386 付属のソースコードをベースに変更を加えたが、現在は

telnet.91.03.25.tar.Z(1991 年 3 月 25 日版)

を元に開発を行っている。

変更したファイルは commands.c, externs.h telnet.c の 3 つであり、更に telnetx.c というファイルを追加している。

ZMODEM や kermit、transit はフリーソフトウェアとして配布されているものをそのまま利用した。B-plus については UNIX 用に単体で配布されているものがなかった為、同じくフリーソフトウェアの tipxbp の中から必要なライブラリを流用した。

これらのプログラムは現在、

```
gladys.cs.uec.ac.jp:~ftp/pub/wpnc
```

から入手可能である。

3.7 利用方法

このプログラムを用い、実際にパソコン通信のアーカイブサービスを利用してみた。対象としたパソコン通信サービスは、

- NIFTY Serve
- BIX (x25.bix.com)

である。前者は B-plus によるライブラリサービスを、後者は ZMODEM を用いたファイルの転送サービスを用意している。

telnet コマンド自体の使用方法はかわらない、接続ホストとして相手先のホスト名を入力する。

通常のパソコン通信の手順でライブラリサービス利用する。転送モードに入ったところでエスケープを入力し、telnet のコマンドモードに移る。コマンドモードのヘルプは以下のようになる。

```
telnet> ?
(省略)
sz              send files by zmodem
rz              receive files by zmodem
bp              transfer files by bplus
kermit          transfer files by kermit
transit         receive files by transit
prg             call transfer prog
?               print help information
telnet>
```

help メニューの中で最後の 5 個がファイル転送関係のコマンドである。

- sz:ZMODEM プロトコル (転送)

- rz:ZMODEM プロトコル (受信)
- bp:B-plus プロトコル (送受信)
- kermit:kermit プロトコル (送受信)
- transit:transit プロトコル (送受信)
- prg:任意の外部プログラムを起動する

それぞれコマンドと同名の外部コマンドを起動する (prg は除く)。対応するコマンドはユーザの実行 PATH に入っている必要がある。コマンドに与える引数はそのまま対応するコマンドに渡される為 kermit のような両用のコマンドも問題はない。

最後の prg コマンドは第 1 引数に指定したコマンドを外部コマンドして起動する。これは上記の転送プログラム以外のものを利用したり、例えばパソコン通信にログインする為のチャットプログラムなどに利用することが可能だ。

転送プログラムの終了シグナルを受け取ると telnet はコマンドモードに戻る、そのままリターンを送ればそのままサービスを継続することが可能である。

3.8　評価

今回 NIFTY Serve と BIX において転送速度の実験を行った。前者は B-plus を用い後者は ZMODEM を用いた。

パソコン通信	プロトコル	通信速度
BIX	Z-modem	2-3000cps
NIFTY Serve	B-plus	700-1200cps

表 3.1: 転送速度

Z-modem のほうは転送速度を表示するコマンドがないので、大体の目安である。NIFTY Serve の速度が回線速度と比較して遅いのはパソコン通信のホストへの接続が ROAD2 相当となっており何らかの制限が設けられている為と思われる。これに対して BIX は Z-modem 自身の圧縮転送も手伝ってか、遠い回線にも係わらずかなり高速な転送速度が実現されている。

どちらにしてもインターネットの回線の混み具合やホストの負荷などによる変動がかなり存在している。また telnet と転送コマンドとの通信には pty を使っていることや転送プログラムの実装上の問題もあり、telnet を起動したホストの負荷もかなり高い。これについては転送プログラムの改良などである程度の改善が可能である。

3.9 今後の課題

今回の実装では telnetx が起動するプログラムは標準入出力を利用するものなら殆どのものが利用可能である。今後これを利用したパソコン通信への自動ログインプログラムや自動ダウンロードプログラムなどの開発が可能だろう。また GNU emacs から利用できるようなインターフェースの開発などが望まれる。

現在 NiftyServe に関しては perl を使用した自動運転スクリプト群が用意されている。これは telnetx を使ったライブラリの自動登録機能などが用意されている。また Gnu Emacs から利用できるインターフェースも登場しており、これも通信には telnetx を利用している。

現在インターネットから利用可能なパソコン通信サービスは着実に増加している。国内の大手 BBS では Nifty Serve と ASCII ネットが利用可能であり、海外では Compu Serve が利用可能になっている。今後も大手 BBS や草の根 BBS 等の接続も増えていくことが予想される。

インターネット側からこれらの BBS を利用する場合、現在のように個々の BBS に対応するのではなく、もっと統一的に利用可能なユーザインターフェースを設計し提案していく必要があると考えている。