Javaによる仮想端末設計におけるポータビリティ性の検討

納富 一宏[1]・長田 穂高[2]・石井 博章[1]

[1]情報工学科

[2]工学部情報工学科4年

Examination of Portable Design for Virtual Terminal in Java

Kazuhiro NOTOMI[1)], Hodaka OSADA[2)], Hiroaki ISHII[1)]

Abstract

In this paper, we propose the method of new development and offer of application which became possible by using Java. It solves inconvenient of install and time of development at once. A user should just work like the time of browsing a home page. A developer can make application which operates in much environment only from writing one code. As a means to prove whether this method is practical, we created the virtual terminal using Java.

Key Words: Java, telnet, Internet, Web Browser

1. はしがき

Windows95 が登場した頃から、コンピュータに興味を持たない人も「インターネット」という言葉を耳にするようになった。しかし、テレビや雑誌で取り上げられるのは、World Wide Web と電子メールだけである。

最近のコンピュータには、高機能な Web ブラウザや電子メール用ソフトが標準で用意されている。しかし、telnet や FTP(File Transfer Protocol)は使い勝手が悪かったり用意されていないことがある。

インターネットを利用してできることは、他にもある。ネットニュースや telnet などが例として挙げられる。さらに、新しい利用方法を考え出し、実装することもできる。インターネットの利用は World Wide Web と電子メールに限定されない。

もちろん利用者が必要に応じてアプリケーションをインストールすればそれで使えるようになるし、今まではずっとそのようにしてコンピュータは利用されてきた。

しかし、「インターネット」が広まると同時にコンピュータ利用者の幅も大きく広がり、必ずしも扱いに慣れた者だけが触るとは限らなくなった。コンピュータの扱いが不慣れな者にとってインストールは、説明文中の用語の意味がわからなかったり、インストールの手順に不明な点があったり、不安を感じながら行うものなので、敬遠されがちである。

加えて、最近は Windows だけでなく PC/AT 互換機で動作する UNIX が数種類開発されたり、多数のオペレーティングシステムが混在するようになった。開発者にとって、異なるプラットフォームにあわせてプログラムを作り直すのは手間のかかる作業である。

そこで本研究では、これを解決するために、新しいアプリケーションの提供方法を提案する。本手法は、利用者がアプリケーションを用意するのではなく、利用して欲しい側が必要なアプリケーションを提供するものである。

本研究では以上の条件を満たすアプリケーションの実現例として、Web ブラウザ上で動作する仮想端末を試作

した。本稿では、この手法が新しいプログラムの開発・提供方法となり得るかどうかを検討する。

2. telnet プロトコルと仮想端末

2.1 仮想端末

telnet を使うと、利用者はリモートコンピュータの機能や資源をローカルコンピュータ上で利用できる。一般に、telnet プロトコルを用いてリモートコンピュータとやり取りをするアプリケーションは仮想端末と呼ばれる。

仮想端末はどんなに高機能であってもかまわないが、最低限必要な機能はそろえていなければならない。必要な最低限の機能をモデル化したものはネットワーク仮想端末(Network Virtual Terminal、以下 NVT)と呼ばれ、telnet プロトコルではコネクションの両端が NVT であることが前提になっている。

NVT が備えていなければならない最低限の条件の中で最も重要なのは、入力された文字を相手に送り、相手から送られてきた文字を表示する機能である。ここでいう文字とは7ビット ASCII 文字であり、少なくとも US ASCII 文字95種をすべて表示できなければならない。また、改行やタブ、バックスペースなどの画面制御コードも正しく表示できなければならない。

2.2 telnet プロトコル

telnet はリモートコンピュータへのオンラインアクセスを実現するためのプロトコルであり、広く一般に用いられているが、その実装は複雑である。

2.1 で述べたように、仮想端末の機能は、高度な機能については制限されていない。しかし、リモートコンピュータとローカルコンピュータが互いにその機能を持っていなければ、その機能を利用することは不可能である。

そこで telnet プロトコルでは、接続後、互いに利用可能な機能について telnet コマンドを用いてオプション交渉をする。

telnet コマンドについては RFC(Request For Comments)854 に記述されている。telnet コマンドには、画面制御用コマンドや telnet オプション交渉用のコマンドが用意されている。

telnet オプションは、他のプロトコルと異なり、1つの RFC にまとめられておらず、オプションごとに分けられている。1つの telnet オプションは1つの機能をあらわしている。

2.3 仮想端末の構成

仮想端末は、機能面から画面表示部とネットワーク部の2つに分けることができる。

2.3.1 画面表示部

仮想端末の画面は、複数行にわたってテキストを表示できる機能を持っている。

画面には、ユーザーが入力した文字が全て表示されるとは限らない。例えば、パスワード入力の際には画面には何も表示されない。

仮想端末が入力された文字を表示するか否かを判断するには、高機能な画面表示部を作成しなければならない。

telnet オプションの中のエコー機能を有効にすれば、画面表示部は、リモートコンピュータから送られてくる文字を表示すればよい。エコー機能を使うと、入力された文字をリモートコンピュータ判断し、表示すべき文字をローカルコンピュータへ送信するようになる。

また、画面表示部には、キーボードからの入力イベントに反応し、リモートコンピュータへ文字を送る機能も必要である。

2.3.2 ネットワーク部

ネットワーク部分は、送信と受信に分けられる。telnet は1バイトを最小の単位として送受信を行う。

2.2 で示した telnet オプション交渉は、仮想端末の動作に必ず必要である。ネットワーク部は、telnet オプション交渉の機能を実装しておかねばならない。

さらに、画面表示部分と連携し、入力された文字をリモートコンピュータへ送信し、受信した文字を画面に表示する。

2.3.3 その他の構成要素

telnet コマンドと telnet オプションは、1 バイトの数字で表現される。例えば、2.3.1 で説明したエコー機能の値は1である。

しかし、数字のままではプログラム開発時に扱いづらい。そこで、telnet コマンドと telnet オプションが定義されている RFC では、コマンド・オプションの略語も定義している。例えば、エコー機能は ECHO という文字で表され、値は1である。

プログラムの開発においては、コマンド・オプション

の略語と値を定義することは、必ずしも必要ではないが、仮想端末を作成する際の手間を減らすものと考えられる。

3. Javaによるシステム設計

3.1 新しいアプリケーション提供方法の提案

今までは、利用者が必要に応じて、自分の利用しているプラットフォーム用のアプリケーションをインストールしていた。異なるプラットフォームのコンピュータを利用するには、別途アプリケーションを用意しなければならない。

そこで、新しいアプリケーション提供方法を提案する。アプリケーションは、ネットワーク上のコンピュータに用意しておく。利用者は、必要に応じてリモートコンピュータへ要求し、アプリケーションをダウンロードして使えばよい。

3.2 Javaの特徴

Java には他の言語には見られない特徴がある。ここで、本手法で用いる開発言語として Java を選んだ理由を挙げる。

①Web ブラウザで実行できる。利用者はアプリケーションのインストールをする必要はなく、ホームページ(HTML ファイル)を開くだけでよい。ただし、Web ブラウザ上で動作させることができるのはアプレットと呼ばれる Java アプリケーションのみである。アプレットは Applet クラスを継承して作られ、ローカルコンピュータのファイルにアクセスできないなどのセキュリティ上の機能制限がある。

②開発したアプリケーションは、再コンパイルをせずに異なるプラットフォームで動作する。ただし、そのプラットフォームに Java を実行できる環境(Java Virtual Machine 以下 Java VM)が用意されていることと、固有のプラットフォームに依存するようなプログラムを書かないことが必要である。

③Socket や URL などの、ネットワークを利用するクラスが、標準 API(Application Programming Interface)として用意されている。このことから、ネットワークを利用するアプリケーションの開発の負担が減る。

④Java に対応したブラウザは、２０種類以上のプラットフォームに存在する。しかも、最近では標準で用意されていることが多い。このことから、利用者が新たに Java VM を用意せずに済むと考えられる。

以上の４点は、まさしく本手法を実現するために必要な特徴である。

3.3 システム構成

今回作成したクラスを以下に示す。

- Client クラス。Applet(標準 API)を継承。ブラウザーから呼ばれる仮想端末の本体部分。
- ネットワーク部として、StreamListener クラス，StreamWriter クラス，SendList クラス。これらは、上から順に、受信、送信、送受信管理の機能がある。
- 画面表示部の、KeyInputAdapter クラス。このクラスは、キー入力イベントを監視し、サーバーへ入力した文字を送信する機能を持つ。文字を表示する部分は、標準 API の TextArea を用いた。
- TelnetCommands クラス、TelnetOptions クラス。この２つのクラスは、2.3.3 で示した機能を持つ。

今回作成したクラスは、どのようなプラットフォームでも動作するよう、標準 API のみ利用した。

4. システムの評価

4.1 評価手順

今回のシステムを、以下の点について評価する。

1) 仮想端末としての動作。
2) 異なるプラットフォームでの動作。

評価には以下の環境を利用した。

- Windows98 と Internet Explorer4.0
- Windows98 と Netscape Communicator4.06
- Linux2.0.35 と Netscape Communicator4.05

評価は、UNIX へログインし、コマンドを実行することで行った。詳細は 4.3 動作確認で示す。

4.2 動作準備

作成した仮想端末は、Java に対応したブラウザ上で動作する。html 内に以下のような記述を含めることで呼び出すことができる。

```
<applet code=Client.class width=800 height=400>
</applet>
```

このタグを含む html ファイルを開けば、通常の仮想端末として動作する。ユーザー名とパスワードを入力しログインすれば、リモートコンピュータが利用できる。

4.3 動作結果

Windows98 上の Internet Explorer4.0 での動作例を図 4.1 に示す。図 4.1 は、balthasar という名前のリモートコンピュータへ接続後、ユーザー名の入力を待っている画面である。

図 4.1(Windows98+Internet Explorer4.0)

次に、Windows98 上の NetscapeCommunicator4.06 での動作例を図 4.2 に示す。リモートコンピュータへログインが完了し、df コマンドを実行した状態である。

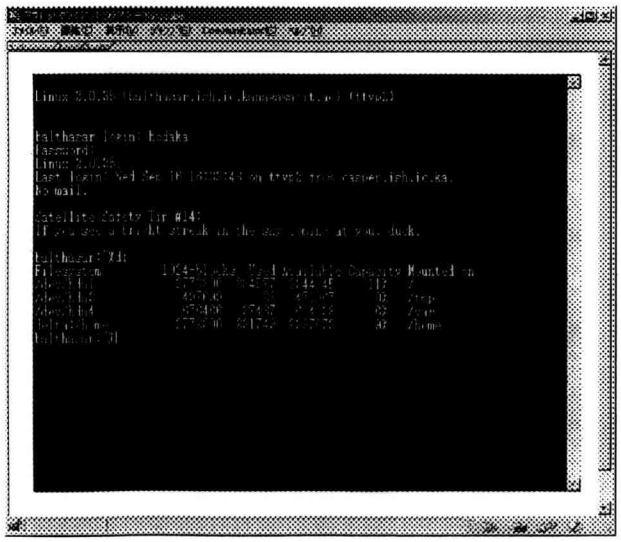

図 4.2(Windows98+NetscapeCommunicator4.06)

次に、ブラウザでなく、オペレーティングシステムを変更した例を図 4.3 に示す。オペレーティングシステムを Linux に変更したが、ホームページの更新や仮想端末の再コンパイルは行っていない。

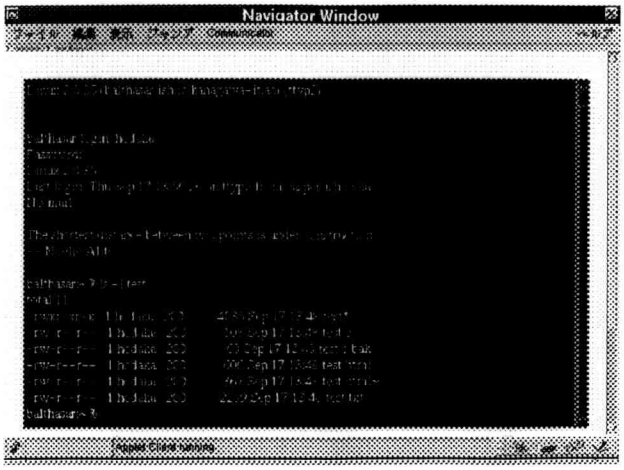

図 4.3(Linux2.0.35+NetscapeCommunicator4.06)

4.4 考察

4.3 の動作結果により、今回作成したシステムが、仮想端末として動作することが確認できた。これは、コマンドが正常に実行され、画面にコマンドの実行結果が表示されている点から明らかである。

また、4.3 の動作結果により、今回作成したシステムが、ホームページの更新もアプリケーションの再コンパイルもせずに、異なるプラットフォームでも同様に動作することが確認できた。これは、4.3 の動作例が全て異なる環境で行われたにもかかわらず、同様に動作していることから明らかである。

5. むすび

今回実装した機能は必要最低限のものであったこともあり、仮想端末としての性能は低い。しかし、ここで注目すべきなのは仮想端末が実現可能であった点である。

本手法による今回の仮想端末の実現は、他の複雑なアプリケーションの実現をも可能にすることを示唆している。他のプロトコルでも、また、ネットワークを使ったアプリケーションでなくとも本手法が有効であるものと考える。

参 考 文 献

1) David Flanagan:『JAVA クイックリファレンス』, オライリージャパン, (1996)

2) Philip Miller: 『マスタリング TCP/IP』, オーム社, (1998)

3) Paul Simoneau: 『TCP/IP プロトコル徹底解析』, 日系 BP 社, (1998)