システムソフトウェアと
オペレーティング・システム 88－9
（２００１．７．２７）

Remote Editing Protocol の Mac OS X のエディタへの応用

宮里 忍† 河野 真治†
†琉球大学工学部情報工学科

概要

ネットワークを介した異なるマシン上に保存されているファイルを編集する既存の方法として、リモートログインや、NFSなどを利用する方法が考えられるが、大きなファイルを扱う場合の通信時間や応答性に問題があった。
本研究では、ファイル編集を行うエディタの機能を分割、双方にバッファを持ち、そのバッファ間で編集内容の通信を行う、リモートエディタが提案されている。
そしてこれまで、Emacs上での実装による通信時間の評価や、編集における応用などについて考案がなされてきた。
本稿では、Mac OS X付属のエディタであるTextEditを用いての実装例を示すと共に、異なるアーキテクチャ上のEmacsとTextEditを接続することが目的である。

Remote Editing Protocol on editor of Mac OS X

†Shinobu Miyazato †Shinji Kono
†Department of Information Engineering, University of the Ryukyus.

Abstract

Remote login or NFS can be used to edit files on network computers. But in case of large file, these methods suffers from response time or large file transfer time. To solve these problems, a remote editor using client server model is shown. The remote editor coordinates both the buffers of the clients and the buffer of the server. It has been designed to evaluate the communication time by mounting on Emacs and to apply for the edit. This paper shows the example of mounting using TextEdit which is the editor of Mac OS X. We connect TextEdit and Emacs which are on different architecture.

リモートエディタとは、サーバ・エディタによってファイル管理を行い、クライアント・エディタを用いてサーバに存在するファイルに接続し編集を行うエディタである。

リモートエディタでは、すべてのファイルをサーバで管理することにより、他のクライアントが編集したファイルに対して追加編集を行うといったことができる。そのため、多人数でのプログラム作成や、書類の編集などが容易になることが期待できる。

これまで、本研究ではEmacs,PICOによるリモートエディタの実装、既存のプロトコルとのネットワークトラフィックの比較評価が行われて来た。

本稿では、Mac OS Xのエディタであるを用いての実装例を示すと共に、TextEditと異なるアーキテクチャ上のEmacsを接続することが目的である。

1 リモートエディタについて

ここでは、まず既存のリモートファイル編集についての方法と問題点を挙げ、次にテキストエディタを例に、通常のエディタとリモートエディタの比較を行うことで、エディタのリモート化による利点について述べる。

1.1 一般的なテキストエディタの機能

テキストエディタでは、メモリ上に編集するテキストを展開し編集作業を行うバッファを用意し、読み込んだファイルをその中へ格納する。編集によるデータの変更はバッファに読み込まれたテキストデータに対して行われる。編集内容の保存はクライアントによる保存操作が行われたときに、そのバッファをファイルとして書き出すことで行われる。

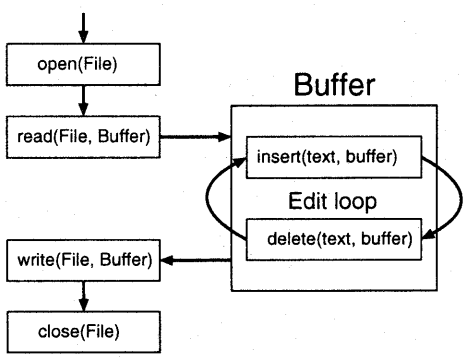

図 1: テキストエディタにおける編集の流れ

1.2 リモートログインを介したファイル編集

リモートログインを用いたファイル編集では、ローカルマシンはキーボード入力のたびに、そのデータをリモートホストに対して送る。リモートホストは受け取ったキー入力に対する処理を行い、その結果をクライアントの端末に返す。現在のネットワークの速度では、その遅延を感じることはまれであるが、ネットワークの速度が極端に低下したときは、キー入力の度に大きな遅延が起こる。また、完全にネットワークが切断されたときは、編集作業は完全に中断されてしまう。

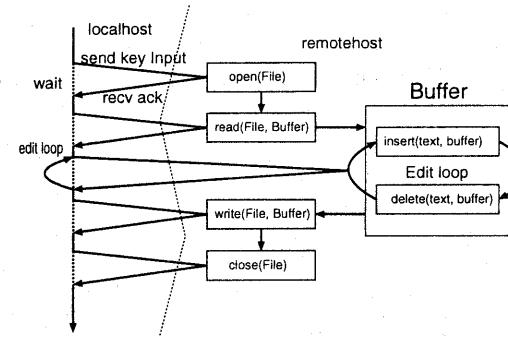

図 2: リモートログインによるファイル編集

1.3 NFS を介したファイル編集

NFS を介した編集では、NFS クライアントはファイルの内容を一度全て受け取り、その受け取ったデータに対して編集作業を行う。そして、保存するときは再びそのデータを NFS を介しサーバに渡され、サーバ上のディスクへファイルを書き込む。この場合、ファイルサイズが小さいときは問題ないが、大きなファイルを編集するときには、そのデータを全て転送することになり、ネットワークやクライアントに対する負荷が大きくなることになる。

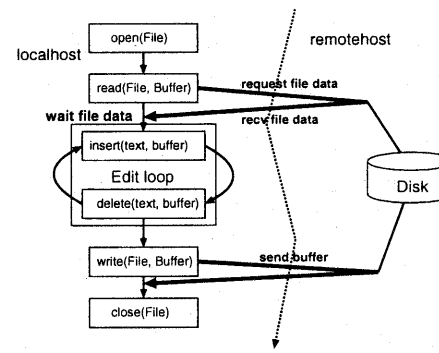

図 3: NFS を介したファイル編集

1.4 リモートエディタ

リモートエディタでは、従来のエディタの機能を、クライアントとサーバの二つに分割し、互いに

通信することでデータの編集・管理を行う。サーバ側では主にファイルと編集バッファの管理を行う。クライアント側は、サーバに接続し編集を行う。

開いたファイルのバッファをサーバ側で管理することにより、NFSのように場合によっては大きなデータを送らなければならない、といったことがなくなる。また、クライアントにバッファを持つことにより、ネットワークの切断が起きても、クライアントのバッファを用いて編集作業を行い、ネットワークが回復したところで変更箇所をサーバに送るようにすることで、編集作業の中断を最小限におさえることができる。

また、通常のエディタでは、編集されたファイルを他のクライアントが編集を行うには、そのファイルのアクセス権限の変更や、一旦自分のカレントディレクトリにコピーして、そのファイルを編集するなどの作業が必要になる。

リモートエディタでは、すべてのファイルをサーバで管理することにより、他のクライアントが編集したファイルに対して追加編集を行うといったことができる。そのため、多人数でのプログラムの作成や、書類の編集などが容易になることが期待できる。

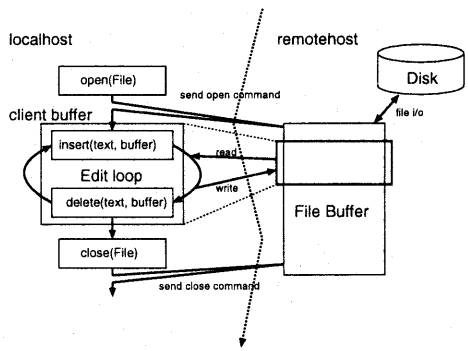

図 4: リモートエディタでのファイル編集

1.5 テキストの編集に必要な要素

テキストエディタでファイルを開いた場合、ファイルの内容は全て、メモリ上にバッファリングされる。このうち、ユーザが編集の際に必要な情報は、画面に表示する分のデータである。

テキストエディタでの編集において、ユーザの直接入力による変更は、画面に表示されている部分にしか行われない。また、テキストエディタには、入力文字列の検索・置換機能が備わっていることが多い。文字列の置換も文書編集機能の一部ではあるが、これはユーザ側には見えない部分での編集操作である。

つまり、ネットワークを介した、エディタによる編集では、クライアント側で行うバッファリングは、画面表示分だけ行うのが最適であると考えられる。

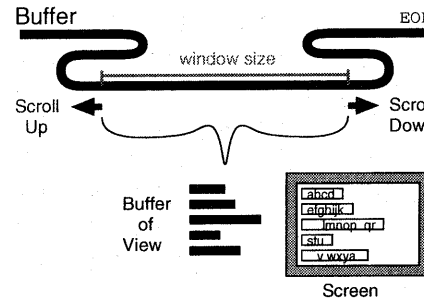

図 5: リモートエディタにおけるデータ転送

1.6 プロトコル

クライアントからのパケットは、1バイトのコマンド識別子に、引数が続く形式である。

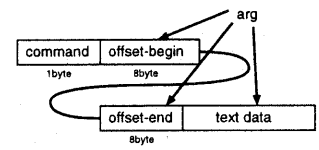

図 6: パケット形式

次に編集コマンドを示す。ここでのサーバとは、編集対象となるバッファを持つエディタ、クライアントとは、そのバッファへ接続し編集を行うエディタのことである。

編集コマンドは主に open、save、close、read、write、update からなる。

- open：サーバ側のファイルの読み込みを行う。引数としてファイル名を与える。サーバ側でファイルが開かれるとクライアントへ、その

バッファ名を、コマンド識別子を付加して返す。また、同時に次の read を行う。

- save：サーバエディタ側のバッファを実際にファイルに書き出す。ファイルはサーバ側のホストに書き出される。
- close：編集バッファを破棄する。サーバはクライアントとのコネクションを切断する。
- read：クライアントで open コマンドが実行されたとき、またはエディタの行移動などで、画面が更新される際に、必要なテキストデータをサーバへ要求する。引数として、要求するテキストのオフセット値を与える。サーバはこのコマンドを受けると、オフセット値から適当な長さのテキストを、コマンド識別子を付加してクライアントへ返す。同時にサーバでは、クライアントへ送ったテキストに対応するバッファのオフセット値を記憶する。

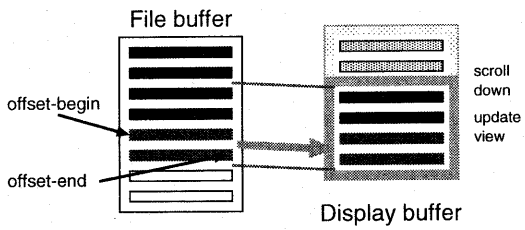

図 7: read コマンド

- write：サーバのバッファに対し、クライアント側で行った変更を加える。引数として、テキストデータおよびそのオフセット値を与える。サーバはこのコマンドを受けると、サーバ側のバッファにおけるオフセット値と、受け取ったデータに含まれる引数のオフセット値を元に、現在のバッファ領域のテキストを受け取ったデータに置換する。そしてサーバは、クライアントに対して、新しいオフセット値を返し、そのオフセット値を記憶する。
- update：複数ユーザによる単一バッファに対しての同時編集時に、サーバ側のバッファに対して行われた変更を、他のクライアントエディタに反映させる。このコマンドは、ある

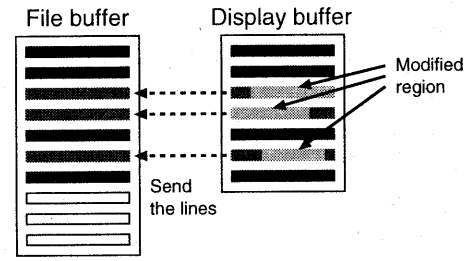

図 8: write コマンド

クライアントによって編集された行の番号とそのテキストを引数として受け取り、それを同じバッファを編集中の他のクライアントエディタへ送る。クライアントはこのコマンドを受け取ると、指定された行番号のテキストを受け取ったテキストデータに置換えする。

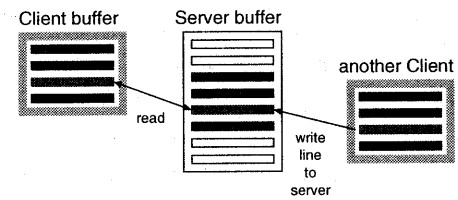

図 9: update コマンド

2 設計、実装

ここでは、まずこれまで行われている Emacs での実装例を紹介し、次に本稿の実装で用いた TextEdit と、それを用いた設計および実装例について述べる。

2.1 クライアント・サーバ型リモートエディタ

ここで紹介する実装例は、一つのエディタをサーバとし、そこへ複数のクライアントエディタを接続する。ファイルの入出力および管理はサーバ側で行う。

2.1.1 Emacs による実装例

Emacs は、Emacs-Lisp という言語処理系を内蔵しており、Emacs-Lisp を用いて、Emacs 自身から別プロセスを起動し、そのプロセスと通信を行うことができる。

Emacs の実装では TCP/IP を用いたクライアント・サーバ型のメッセージ送受信プログラムを作成し、それを Emacs 内部で実行することで、エディタ間の通信を行っている。

また、Emacs は標準でマルチバッファの機能を備えている。各々のクライアントが個々のバッファを持つようにすることで、ネットワーク切断時の再接続においても、クライアントは切断前の編集状態を維持することができる。そのことから、1 つのバッファを 1 クライアントに対応させるという実装になっている。

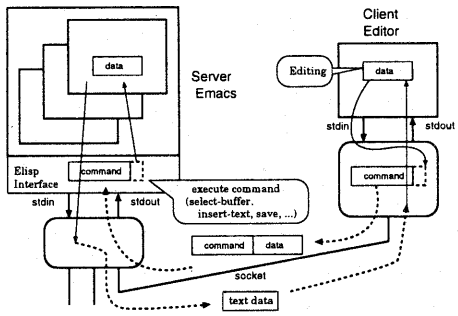

図 10: Emacs におけるリモートエディタ

2.2 TextEdit による実装

ここでは、本稿の実装で用いた TextEdit の特徴と、リモートエディタとしての設計および実装例について述べる。TextEdit は Mac OS X に付属しているエディタで、通常のプレインテキストエディタの他に RTF(RichTextFormat)、HTML エディタとして使うことの出来る多機能なエディタである。Objective-C により記述されたものと、Java により記述されたものがあるが本実装では Objective-C で記述された方を用いて実装を行った。リモートエディタを作成するにあたり、TextEdit はリモートエディタサーバとし、クライアントには前節で述べた Emacs を用いた。

2.2.1 リモートエディタサーバとしての TextEdit

TextEdit にはクライアントエディタとのメッセージ送受信を行う communicater と、クライアントから送られるパケットを展開処理するための tokenizer を実装した。TextEdit と Emacs 間でやりとりするテキストデータの文字コードは EUC に統一した。本来 TextEdit では EUC を扱うことは出来ないが、上記の理由から EUC を扱えるように改変を加えてある。

- communicater: TCP/IP を用いたメッセージ送受信インターフェースであり、クライアントの接続の受け付け、ソケットの管理を行い、クライアント側で起動されるクライアント通信インターフェースとの間にエディタ間のデータストリームを提供する。また、クライアントからのメッセージを tokenizer に渡すと共に、tokenizer から受け取るメッセージをクライアントへ送る。
- tokenizer: クライアントからのメッセージを展開処理し、TextEdit のエディタ部分へ相当する処理を渡す。また、クライアントへのメッセージを生成し communicater に渡す。送受信するテキストデータの文字コード変換もここで行う。

TextEdit では、クライアントによって開かれるファイル全体のバッファとバッファ名、及びクライアントへ送信したバッファ領域へのポインタが管理される。以上の実装で、TextEdit と異なるアーキテクチャ上の Emacs をエディタレベルで接続することができた。

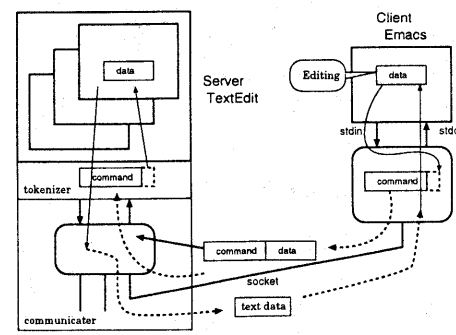

図 11: TextEdit-Emacs によるリモートエディタ

3 考察

3.1 Mac OS X におけるその他の実装方法について

前章で述べたTextEditでの実装のようなエディタ自身にリモートエディタとしての機能を付加する方法の場合、実装元となるエディタの仕様を少なからず変更せざるを得ない。

そこで、Mac OSにおけるタスクのコントロール・自動化を行うツールとして提供されているAppleScriptを用いてリモートエディタサーバを実現する方法を考える。

3.1.1 AppleScriptの概要

AppleScriptはオブジェクト指向スクリプト言語であり、Mac OSにおいてユーザが行う処理の殆どがAppleScriptに置き換えることが出来る。AppleScriptはユーザから与えられたスクリプトファイルを元に、AppleEventと呼ばれる独自のメッセージ交換ツールを使い、Mac OSアプリケーションの操作・コントロールを行う。

3.1.2 communicaterとtokenizer

AppleScriptを用いた実装においても前章で述べたような communicater,tokenizer が必要になる。しかし、AppleScriptのみではそれらの機能を実現するのはAppleScriptの仕様上難しい。そのことから、communicater,tokenizerはAppleScriptとは別に作成し、AppleScriptからそれらを参照するというのが最適であると思われる。ここでのcommunicater,tokenizerは前章で述べたものと同等のものであるが、tokenizerはサーバエディタに処理を渡すのではなく、AppleScriptに処理を渡す。

3.1.3 AppleScriptで行う処理

AppleScript側ではtokenizerから処理を受け取り、実際にエディタに対してその処理を行う。また、クライアントへのメッセージをtokenizerに渡す処理も行う。このような方法を用いるとEmacsでの実装と同様に実装元のエディタを改変する必要は無く、エディタの仕様が変わることは無い。また、サーバ側で用いるエディタはAppleScript側のみで管理されるので、比較的容易にサーバエディタを変えることが出来るのではないかと期待できる。さらには、AppleScriptはリモートホスト上のMac OSアプリケーションを操作することが可能であるので、一つのホストにリモートエディタサーバを実装することで複数のホストのファイルを編集することが可能になる。しかし、サーバ側のエディタがAppleScriptから操作可能である必要があるという問題もある。

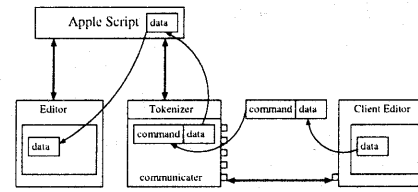

図 12: AppleScriptを用いたリモートエディタ

3.2 Simple Object Access Protocol

ここではDevelopMentor、IBM、Lotus、Microsoft、UserLandSoftwareにより提案されているSimple Object Access Protocol(SOAP)の概要を説明し、次節でRemote Editing Protocolとの比較を行う。

3.2.1 SOAPの概要

SOAPは非集中／分散環境におけるシステム間の構造化され型付けされた情報の交換のためのXMLを用いた単純で軽量なプロトコルである。SOAP自体はプログラミングモデルや実装固有の意味といったものを定義せず、データをモジュールとして符号化するためのモジュール式のパッケージングモデルと符号化のメカニズムを提供することにより、アプリケーションのセマンティクスを表現する単純なメカニズムを定義する。大きく分けると次の3つの要素から成り立っている。

- Envelope構成要素
  - 何が(what)メッセージの中にあるのか、

表 1: REP,SOAP,AppleScript の比較表

	REP	SOAP	AppleScript
位置付け	プロトコル	プロトコル	言語
通信単位	編集データ	SOAP メッセージ	AppleEvent
用途	異なるアプリケーション間での相互編集	分散環境におけるシステムの連係	Mac OS アプリケーションの操作・コントロール
汎用性	○	○	×

誰が (who) それを処理すべきなのか、それは選択可能か必須かどちらなのか (whether) といった表現をするための全体の枠組みを定義する。

- Encoding(符号化規則)
  - アプリケーションが定義したデータ型のインスタンスをやりとりするための直列化のメカニズムを定義する。
- RPC 表現
  - RPC(Remote Procedure Call) とそのレスポンスを表現するための規約を定義している。

3.3 比較

Remote Editing Protocol と、これまでに述べた SOAP、AppleScript とを特徴と位置付けの面から比較し、表 1 にまとめる。

- Remote Editing Protocol :

  エディタの編集に用いられる一般的な機能を統一し、それを用いて異なるアプリケーション間で直接通信を行い、相互編集を行う機能を提供する。これは編集データを単位としたプロトコルであるということである。クライアント・サーバ間で編集データをやり取りすることで、クライアントはサーバ上に存在するファイルをモディファイすることが出来る。編集データを単位とした単純なプロトコルとすることで汎用性を持たせている。

- SOAP :

  SOAP はまず、送信側は符号化規則に則って送信データをエンコードし、受信側に送る。受信側はそのエンコードされたデータを符号化規則を元にデコードして実行する。実行後必要であれば送信側へレスポンスを返す。この手続きを相互に行うことで異なったアプリケーション間の連携を実現する。これは XML を用いた SORP メッセージを単位としたプロトコルであるということである。SOAP メッセージの符号化を XML を用いたテキストベースにすることで汎用性を持たせている。

- AppleScript :

  まず上の二つと違うところは、AppleScript は言語であるということである。AppleScript は内部的に AppleEvent と呼ばれるメッセージ交換ツールを使ってローカル又はリモート上の Mac OS のアプリケーションにメッセージを送り、そのアプリケーションの操作・コントロールを行う。Mac OS においてはこの方法でリモート上のファイルをモディファイすることも可能であるが、AppleEvent は Mac OS でしか使用することが出来ないので汎用性は低いと言える。

4 課題

4.1 Remote Editing Protocolの適応性

本稿での実装により、テキストエディタに対してプロトコルを適応する手間が、ベースとなるエディタの設計に大きく依存することが分かった。そこで、現存するエディタへのプロトコルの適応が容易であるようなAPIや、軽量で可搬性のあるクライアントエディタを提供することが必要であると考える。また、Remote Editing Protocolをテキストエディタ以外のアプリケーションに適応させていくことも課題である。

参考文献

[1] リモートエディタの実装とそのXMLへの応用 新垣将史, 河野真治 日本ソフトウェア科学会第16回論文集,1999,pp293-296

[2] Emacs上のリモートエディタ 新垣将史, 河野真治 電子情報通信学会技術研究報告,2000

[3] RemoteEditingProtocolを用いた複数ユーザ編集システム 新垣将史, 河野真治 日本ソフトウェア科学会第17回論文修,2000

[4] Apple Developers Connection "Cocoa Developer Documentation" http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html

[5] David A.Curry著、アスキー書籍編集部監訳 "UNIX Cプログラミング" アスキー 1991.

[6] W.Richard Stevens著、篠田陽一訳 "UNIXネットワーキングプログラミング" トッパン 1992.