Remote Editing Protocolを用いた複数ユーザ編集システム

新垣 将史† 河野 真治†
†琉球大学理工学研究科情報工学専攻

概要

リモートエディタとは、ファイルの編集をネットワーク越しに、専用のプロトコルを用いて行うものである。これまで、エディタの機能をサーバとクライントに分割し、双方にバッファを持たせ、そのバッファ間で編集内容の通信を行う、プロトコルを提案してきた。本稿では、Emacs 上でのリモートエディタの実現方法を述べる。さらに、ユーザ間のテキストの共有や、複数のユーザによる同時編集などの問題点を指摘し、それを解決するための効率的な編集のトランザクションについて考察を行う。

1 リモートエディタ

リモートエディタは、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、専用のプロトコルを用いて編集を行う。リモートエディタの操作は、サーバ側で管理するファイル全体のバッファと、クライアントエディタのバッファ間で行われる。リモートエディタには、既存の編集方法 (リモートログイン、NFS など) に比べ以下のような利点がある。

リモートログインを用いたファイル編集では、端末のキー入力、画面出力とサーバの間でリアルタイムの通信が行われるので、ネットワーク速度がユーザの編集作業に直接影響を与えてしまう。リモートエディタでは、エコーバックや、変更の確認などは、ローカルなクライアント上で行うので、ユーザへの応答がネットワーク速度に影響されにくい。

NFS を介した編集では、ファイル入出力の際に、編集するテキストデータを全てネットワークを通して送る必要がある。したがって、大きなファイルを編集する時にネットワークの負荷が大きくなる。一文字挿入する場合には、リモートエディタでは、挿入コマンドを送るだけだが、NFS の場合は大量のデータをローカルなエディタと NFS ファイルシステムの間でやりとりすることになる。

また、リモートログインや NFS の場合は、ネットワーク故障の際には作業が完全に停止してしまうが、リモートエディタの場合は、ローカルに持っているバッファ内の変更に対しては影響がない。また、リモートログインでは不可能な、再接続時のリカバリが可能である。

2 テキスト編集の流れ

一般的なテキストエディタでは、読み込んだファイルの内容を、バッファを用意しその中へ格納する。編集による変更は、バッファに読み込まれたデータに対して行われる。そしてエディタの終了時に編集内容の保存は、バッファをファイルに書き出すことで行われる。

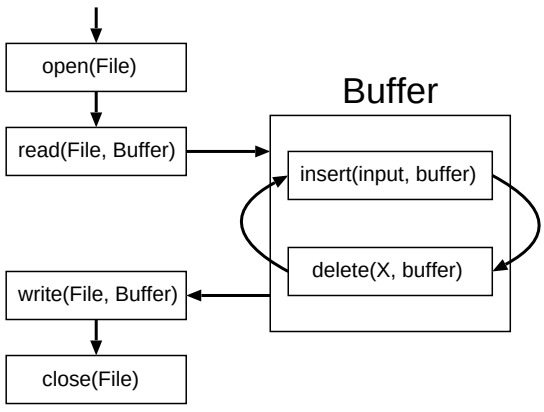

図 1: 編集の流れ

2.1 編集に必要なテキストデータ

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。

2.2 マルチユーザによる同一ファイルの編集

ある既存のテキストに対して、複数のユーザが同時に編集を行う場合について考える。

あるユーザが編集中のテキストに対して、別なユーザにより同一のテキストファイルの保存が行われたとき、書き込まれたデータがまだ編集中のユーザに対して、妨げにならないようにしなければならない。

また、各ユーザによる変更は別々に保持するのが望ましい。

そこで本システムでは、複数の変更が同時にあった場合、テキストのバージョン分岐を行うようにする。

ファイルのバージョン管理を行うツールに CVS がある。CVS では、管理対象のファイルについてバージョンアップ処理を行うと、その変更は、ファイル管理ディレクトリ内に、前バージョンとの差分を記したファイルとして保存する。そして、異なるバージョン間の結合 (マージ) は、ツールによって示されるバージョン間の差分を、手動で変更を加えることにより行われる。

リモートエディタでは、CVS のバージョン管理とマージに似た処理をサーバ内部で行う。

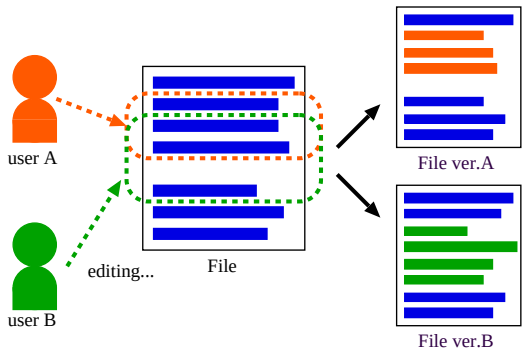

図 2: テキストのバージョン分岐

2.3 テキストのマージ

分岐した別バージョンを 1 つに結合する作業をマージとする。

マージの作業は、編集中のバージョンに、マージしたいバージョンを指定することでその差分を取り込み、それを手動で修正することで行う。

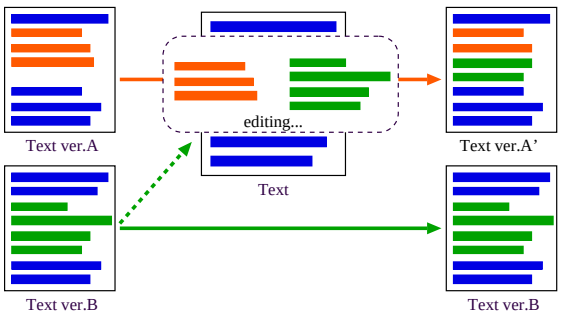

図 3: テキストのマージ

マージ後も、編集 (マージ) したバージョンと差分を取り込んだバージョンの両方を保存する。

3 Emacs 上の実装

ここでは Emacs エディタの特徴を、特に実装上重要な点について挙げる。

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。

また、Emacs は標準でマルチバッファの機能を備えている。そこで、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。

また、Emacs では、様々なイベントごとにフックと呼ばれる変数が存在する。フックには Lisp 式や関数名をリストとして入れておくことで、イベントが起きた際に、その Lisp 式、関数が評価される。

実装に用いたフックを次に挙げる。

- window-scroll-functions

  画面の更新時に呼び出されるフック

- before-change-functions

  バッファに変更が加えられる直前に実行される

- after-change-functions

  バッファに変更が加えられた直後に実行される

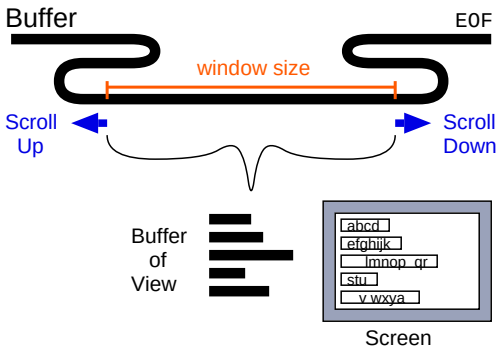

図 4: 通信する編集データ

4 設計、実装

実装に用いるリモートエディタサーバのベースには Emacs を用いる。そして、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う。

サーバエディタでは、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。

サーバ通信インタフェースは、クライアントの接続の受け付けおよびソケットの管理と、クライアント側で起動される、クライアント通信インタフェースとの間に、エディタ間のデータストリームを提供する。

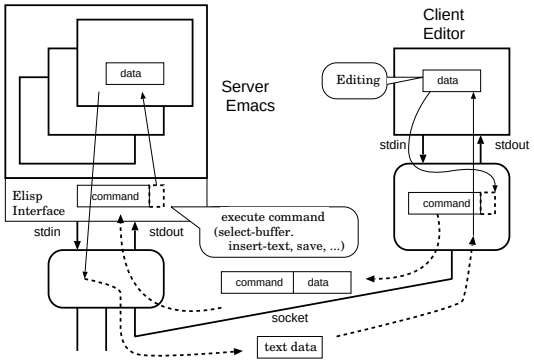

図 5: リモートエディタ

クライアントエディタでは、編集に必要な、画面表示分のバッファを持ち、また、その領域のサーバ側に開かれたファイルにおけるオフセット値を保持する。そして、テキスト内での移動などで、画面の更新が行なわれる際には、再びサーバへ表示に必要なテキストデータを要求する。

また、設計方針として、Emacs 本体をなるべく改変せず、Emacs 側のインタフェースはできるだけ Emacs-Lisp を用いた。

4.1 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、引数が続く形式とした。

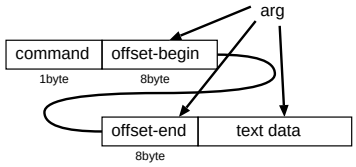

図 6: パケット形式

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とする。

テキストの編集は、基本的に、open、read、write、close、の 4 つから成る。

- 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。

- 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。

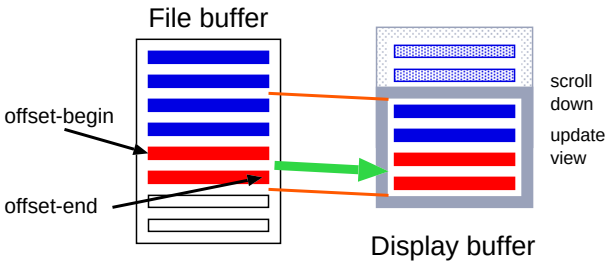

図 7: read コマンド

- 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、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。本実装において、write コマンドは、クライアントの画面更新の際に、クライアントビューから外れた領域において更新があった場合、その領域に対して実行される。

- save: サーバ側のバッファを実際にファイルに書き出す。書き出し後、複数バージョンが存在する場合はマージを行うかどうか問い合わせ、行う場合は、そのバージョンの差分データを取り出し、マージを行う。

- 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。

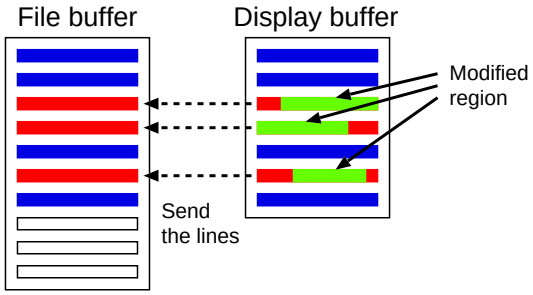

図 8: write コマンド

5 課題

クライアントで用いるエディタが選択できれば、ユーザに対して様々な操作、環境を提供できる。クライアントは、ユーザに対して画面とコマンド (キー操作) を提供するものであり、サーバで行われている処理は気にしなくてもよい。

異なるエディタ間の通信では、編集を行う関数に対して与えるデータ形式を統一しなければならない。

本稿では、どのエディタでも最低持っているであろうテキストの挿入、削除コマンドと、テキストデータのオフセット値を用いた方法を紹介した。

これまで実装では、クライアントエディタとクライアント通信インタフェース間の通信および、パケットの展開処理に、Emacs-Lisp を使用している。この部分に相当するプログラムを、エディタごとに作成することで、通信が可能になると考えられる。

参考文献

[1] Hal Stern 著、倉骨彰訳、砂原秀樹監訳 ”NFS & NIS” アスキー 1992.

[2] 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.

[3] 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.

[4] ”Current Version System” http://www.cvshome.org/

[5] 新垣将史、河野真治 ”リモートエディタのプロトコルとその XML への応用” 第１６回日本ソフトウェア科学会大会論文集 1999.

[6] 新垣将史、河野真治 ”Emacs 上のリモートエディタ” 電子情報通信学会技術研究報告 2000/05/25.