アプリケーション間協調のための遠隔双方向編集プロトコル

Bi-directional Remote Editing Protocol for Application Cooperation

宮里 忍　琉球大学理工学研究科情報工学専攻
河野 真治　琉球大学情報工学科

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

複数のアプリケーションが協調して動作する場合には、双方向の通信が必須である。協調動作に使用するデータをテキストとして表現すれば、その協調動作はテキストを遠隔双方向編集することで実現できる。このために、我々が提案しているリモート編集プロトコルを使用する。本論文では，双方編集プロトコルにリング状のネットワークトポロジーを用いることで負荷が中央に集中することを避けると共に、接続されているアプリケーションが共有する状態を保持する方法を提案する。さらに、複数のアプリケーションを管理するためにセッションマネジャーを Emacs 上で実装する方法を提案する。

1　はじめに

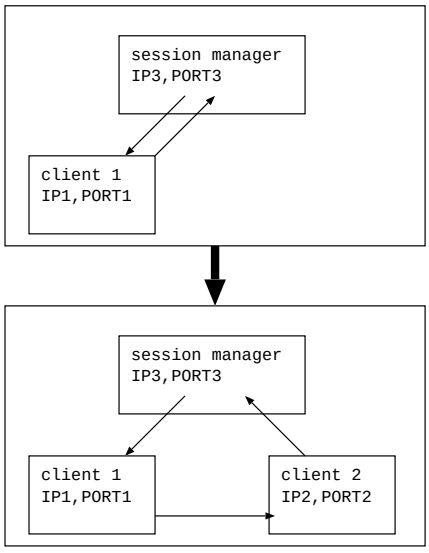

図 1: セッションマネージャを用いたセッションへの参加

PC 上の様々なアプリケーションは、単独に動作するだけでなく複数のアプリケーションが連係して動作することが多い。Office や Browser なども複数のアプリケーションの集合として動作している。しかし、アプリケーション相互の通信プロトコルはカット・アンド・ペーストのような原始的なものか、マイクロソフトの COM[1] のような極めて複雑なもののどちらかしかない。そこで、より理解しやすいアプリケーション間の協調モデルが必要であると考えられる。

我々はこれまで、Pico,Emacs,Mac OS X の標準テキストエディタ TextEdit, 同じく Mac OS X のベクトル系ドローツール Sketch 上でリモートエディタの実装を行い、リモートエディタ間のテキスト編集に特化したネットワーク・プロトコルとしてリモートエディティング・プロトコル (REP) [3][4][5][6][7][8] を提案してきた。REP コマンドには open,save,close,read,write,update コマンドがあり、それぞれテキストエディタにおける処理に対応している。REP は行思考のプロトコルとして設計されており、バッファの read,write は行単位で行われる。サーバおよびクライアントに存在するテキストバッファを操作するコマンド列によって構成された REP は、ネットワークの遅延、切断に強く、巨大なテキストファイルの編集を可能にする。

SOAP や AppleScript は、アプリケーション間を結ぶメソッドを提供しているが、その機能は、特別な機能を呼び出すというものであり、「任意の二つのアプリケーションを接続する」ようなものとは異なる。REP は、「テキストを編集する」という汎用の機能を提供しており、そのテキストの中身に関して REP が知識を持つ必要はない。したがって、編集機能を提供するというインタフェースを提供するアプリケーションがあれば、そのアプリケーションに REP の機能を付加することは容易である。また、意図的に、「テキスト編集による機能の制御」を持つアプリ

ケーションを作成することにより、REP によって相互に接続可能なアプリケーション群を作成することができる。

本研究ではサーバ・クライアント型の接続形態を用い、アプリケーション間をネットワーク上で接続してきた。本論文では、セッションマネージャ図 1 を介してリング状のネットワークを形成することで中央集中型のネットワークを用いずに、接続されているアプリケーションが共有する状態を保持する方法を提案する。例として，複数のテキストエディタをリング状に接続し、それぞれの編集をリモート・エディティング・プロトコルを用いて巡回させ、全体の編集状態を保持し、テキストエディタ間で双方向の編集を可能にする手法を示し、Emacs 上での実装を行う。

2 セッションマネージャ

セッションマネージャは編集に参加しているエディタが存在するマシンの IP アドレス, ポート番号, バッファ名をセッションバッファとして保持し、現在行われている編集を管理する。クライアントはセッションマネージャに接続し、セッションマネージャが動いているマシン上に存在するファイルを開き、編集を開始する。セッション全体としてのバッファの状態もセッションマネージャに保持され、最終的にファイルに書き出される。

編集に参加したい場合は、まずセッションマネージャに接続し、セッションバッファを取得した後に、セッションバッファに IP アドレス, ポート番号, バッファ名を書き込む。書き込まれた情報は REP を用いてセッションマネージャ上のセッションバッファに反映され、そのセッションバッファをセッションに参加しているエディタ間で共有し、同一のセッションに属するエディタ同士がリング状のネットワークを形成する。

2.1 セッション参加要求とリング形成の流れ

図 1 のように Client2 が新たにセッションに加わる際の手順を以下に示す。

1. Client2 がセッションマネージャに接続
2. Client2 がセッションバッファをセッションマネージャから取得
3. Client2 が参加したいセッションバッファに書き込む
4. Client2 がセッションバッファへの変更をセッションマネージャに送る
5. セッションマネージャは新たなセッションバッファを Client1 に送り、Client2 にセッションへの参加が成功したことを通知する
6. セッションバッファを元に Client2 は Client1 からの接続を待つ
7. セッションバッファを元に Client1 が Client2 へと接続を切替える

```
Session Buffer
session1
 IP1,PORT1,buffer_name
 IP2,PORT2,buffer_name
 IP3,PORT3,buffer_name
session2
 ............
 ............
```

図 2: セッションバッファ

セッション参加の手順は始めにセッションマネージャに接続して行われるが、セッションマネージャはセッションバッファをクライアントに送ることと、クライアントからのセッションバッファへの書き込みをセッションバッファに反映し、その書き込みを別のクライアントにも送るということを行っている。このセッションへの参加要求を受け付ける処理はセッションマネージャ特有の機能ではなく、リモートエディタであれば可能な機能であり、既にセッションに参加しているクライアントであればセッションバッファを持っているのでセッションマネージャと同様に振舞うことができる。よって、図 1 で Client2 はセッションマネージャに接続しているが、Client1 に接続してセッションに参加することもできる。

3 行変換テーブル

図 1 の Client1,2 のように、同一のセッションに属して相互に編集が行われている場合、行の挿入・削除が行われたときに、Client1 が意図する行番号と Client2 が意図する行番号とで不一致が起こる可能性がある。Client1 の編集を Client2 に反映する場合は、Client2 が行った編集を考慮した上で編集を反映させなければならない。そこで、Client1,2 は自らが行った編集を行変換テーブルという形で保持し、他のエディタから送られてくる編集が示す行を自バッファ

の行に変換する。また、他のエディタから送られて来る編集を自バッファに反映させない場合にも、行変換テーブルにその情報を書き込む。そうすることで、実際に編集を行ったエディタは他のエディタの状態を気にする必要がなくなる。

3.1 行変換テーブルの更新と役割

行変換テーブルは以下のタイミングで更新される。

- 行の削除・挿入が行われたとき
- 自分が送出した編集メッセージがリング内を一周して戻って来たとき
- 他エディタからの編集を自バッファに反映させないとき

行変換テーブルは他エディタが持っているバッファと自バッファとの違いを吸収するだけでなく、例えば Client1 が編集を終了しようとした時点で行変換テーブルが空でないならセッションマネージャ上に存在するバッファの内容と違いがあることになるので、行変換テーブルをパッチのように用いセッションマネージャ上に Client1 バージョンのファイルを作成するといったことができる。

3.2 セッション内の編集の流れ

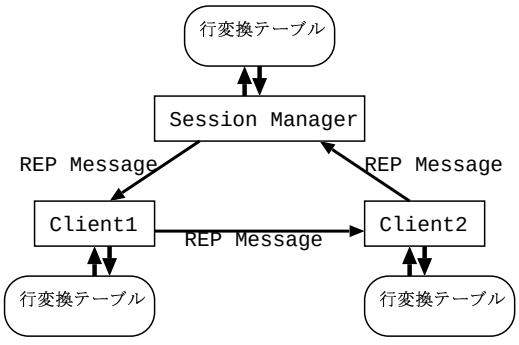

図 3: REP メッセージの流れ

図 3 の Client1 が編集を行った際に起こる REP メッセージの流れを以下に示す。

1. Client1 が自分が持っているバッファに編集を行う
2. 編集内容を REP にエンコードし Client2 に送る
3. Client2 では REP をデコードし、行変換テーブルから Client1 が行った編集の行番号を得て、自バッファに編集を反映する
4. Client2 は Client1 から受け取った REP メッセージをセッションマネージャに送る
5. セッションマネージャも Client2 と同様の処理で自バッファに Client1 の編集を反映する
6. セッションマネージャは Client2 から受け取った REP メッセージを Client1 に送る
7. Client1 は自分で送ったメッセージがリング内を一周し戻って来た場合、そのメッセージを破棄する

4 Emacs 上での実装

ここまで説明してきたセッションマネージャ、行変換テーブルの Emacs 上での実装について述べる。

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。

Emacs における実装では TCP/IP を用いた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。行変換テーブルもメッセージ送受信プログラムの中に実装し、エディタとは別に管理される。

セッションバッファは Emacs にバッファとして持ち、セッションバッファに対する編集は Emacs-Lisp を用いて行う。

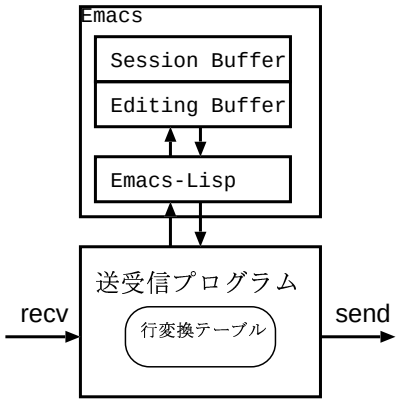

図 4: Emacs 上の実装

4.1 リングネットワークの形成

送受信プログラムは受信用ソケットと送信用ソケットの二つのソケットをセッションバッファの内容に従って開く。セションバッファが図 2 のような状態

のときは、session1に属するClient2(IP2,PORT2)は自分の一つ上の行のClient1(IP1,PORT1)からのメッセージを受け付け、自分の一つ下の行のセッションマネージャ(IP3,PORT3)に対してメッセージを送信する。同一のセッションバッファに対して、それに属するクライアントが送受信ソケットを開くことでリング状のネットワークが形成される。

新しいクライアントが参加しようとしたときや、セッションからクライアントが抜ける場合にはセッションマネージャが更新されるので、セッションに参加しているクライアントはセッションバッファを常に監視し、自分の上か下の行が更新された場合にはそれに従って送受信ソケットを設定しなおす。

セッションバッファはsessionという名前のバッファとし、クライアントがセッションへ参加するときはREP の open コマンドで session という名前のバッファを開くことでセッションバッファを取得する。送受信プログラム側からセッションバッファへの書き込みかどうかを知りたい場合はREP メッセージをエンコードしバッファ名が sessionかどうかで判断することができる。

4.2 行変換

他のエディタから受信した REP メッセージはデコードし行変換テーブルを用いて行番号を自バッファの行番号に変換した後に、送受信プログラム側からEmacs-Lisp に渡され、Emacs 上のバッファに反映される。Emacsにおいて編集が行われると、Emacs-Lispがそれを検知し、編集の内容を REP メッセージにエンコードして送受信プログラムに渡される。送受信プログラム側では、受け取った REP メッセージを行変換テーブルを用いて行番号を変換した後に、他のエディタに送信する。

他のエディタから受信したREP メッセージが、リング内を一周して戻って来た自分が出したメッセージだった場合は、送受信プログラム側で破棄することで、REP メッセージがリング内を周り続けることを防ぐ。

5 まとめと今後の課題

本論文ではセッションマネージャを介し、セッションバッファを共有することでテキストエディタをリング状に接続してリモート・エディティング・プロトコルを用い、複数のエディタ間で編集しあう方法を提案し、Emacs 上での実装方法について述べた。

現在の実装ではリングを形成しているエディタが一つでも切断のプロセスを踏まずに切断してしまうと、その時点で編集セッション全体が止まってしまうという問題がる。セッションに参加しているエディタがネットワーク上に存在しているかどうかを検知し、もし存在していなければセッションバッファを変更しネットワークを再形成するような仕組みが必要である。

また、リング内をメッセージが一周するまでにかかる時間はそれぞれのクライアントが使用しているネットワークの速度に依存し、接続しているクライアントの数に比例して増加てしまう。ネットワークの速度やユーザの利便性を考慮した、セッションの分割やグルーピングのための仕組みが必要である。

参考文献

[1] Guy Eddon, Henry Eddon 著 Inside Distributed COM, アスキー出版局,1998

[2] K. Chandy and L. Lamport: Distributed snapshots: Determining global states of distributed systems. *ACM Transaction on Computer Systems, Vol. 3, No. 1, pp63-75,1985.*

[3] リモートエディタの SVG への応用　宮里忍, 河野真治　日本ソフトウェア科学会第 19 回論文修,2002

[4] リモートエディタの kterm への応用, 情報処理学会第 90 回システムソフトウェアとオペレーティングシステム研究会, 宮里 忍, 河野 真治,200 2.

[5] リモートエディティングプロトコルの Mac OS X のエディタへの応用 宮里忍, 河野真治　SWoPP 2001, July, 2001

[6] 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

[7] 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

[8] 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