リモートエディタの Eclipse への実装

宮城健太[†]　河野真治[††]

これまで我々が提案して来た Remote Editing Protocol (REP) は、異なるマシン上に存在する異なるアプリケーション間を相互に接続する非同期アプリケーションプロトコルである。REP は、Emacs や vim にリモートエディタとして実装されている。今回、新たに Eclipse にプラグインとして実装し、そこではリモートエディタ同士の煩雑な接続手続きを簡略化する。また、それらを制御する SessionManager を実装し評価する。

Implementing of Remote Editor on Eclipse

Kenta MIYAGI[†] and Shinji KONO[††]

The Remote Editing Protocol (REP) we have introduced is a asynchronous application protocol which connect mutually two or more different applications that exist on different machines. REP has been implemented on Emacs and vim. This time, we newly implement remote editor on Eclipse as a plug-in, and simplify complex connection-procedures of the remote editor. In addition, we implement and evaluate SessionManager that controls remote editors.

1. はじめに

これまで我々が提案して来た Remote Editing Protocol (REP) は、異なるマシン上に存在する異なるアプリケーションを間を相互に接続する非同期アプリケーションプロトコルである[1)]。REP をテキストエディタへ実装したリモートエディタは、同一のファイルを複数の異なるマシンから、異なるアプリケーションを用いて編集する事を可能にしたエディタであり、空間的な束縛を受けないため開発者がお互いに密に協調する事ができる。リモートエディタは、これまで Emacs や vim に実装されている。また、REP を既存のアプリケーションに実装することにより、様々なアプリケーションのリモートエディタ化が可能である。

本稿では REP を Java 用の開発環境 (IDE) である Eclipse のプラグインとして実装を行う。これにより、Emacs、vi、と Eclipse を相互に接続することが可能になる。一人で使用する場合でも、IDE では、メソッドや変数名の補完、あるいは、Eclipse の特徴の一つであるリファクタリングの機能を使用することが可能であり、一方で、Emacs や vi などの高度な編集機能を使用することも可能となる。また、XP などの特徴であるペアプログラミングや、先生と学生の作業などの教育的な使い方も可能となる。

今までの実装では、SessionManager へ接続するエディタの指定は、エディタの拡張コマンドを追加し、そのコマンドを入力し、ホスト名やポート番号を入力することにより接続していたが、それではユーザの入力が煩雑になるため、SessionManager の実装を変更し、SessionManager 側の GUI により、接続先を指定するようにした。これにより、一つのマシン内では、リモートエディタはデフォルトの SessionManager に接続するだけなので、ユーザが接続先をエディタ内部から指定する必要はなくなる。従来の SessionManager は Perl で記述されていたが、今回、Java による実装を行う。加えて、Eclipse の GUI を利用することにより、ユーザはマウス操作のみにより SessionManager への接続を行うことができるようになる。

2. 関連研究

2.1 SOBA Project[7)]

SOBA Project は SOBA フレームワークを提案している。開発者は SOBA フレームワークを使用し、ネットワークアプリケーションを作成する。SOBA フレームワークは、インターネット上に仮想的な共有空間を創りだし、ファイルや画像といった静的なメディアから、動画や音声などのストリーミングメディアまで、様々なメディアを共有することを可能とする。

† 琉球大学理工学研究科情報工学専攻
Information Engineering speciality, Science-and-Engineering graduate course, University of the Ryukyus.
†† 琉球大学工学部情報工学科
Information Engineering, University of the Ryukyus.

SOBA フレームワークは P2P 技術を基本として実現されている。

これに対し、REP はプロトコルであり、既存のアプリケーションに REP を実装することにより、そのアプリケーションのリモートエディタ化を実現している。REP はテキスト編集に特化したプロトコルであり、様々なアプリケーションに実装されることにより、ユーザの慣れた環境でテキスト編集を行うことができる。

SOBA Project に対する REP の利点として、既存のアプリケーションの一部を改変するだけでリモートエディタの実現が可能であることが挙げられる。

欠点は、サーバに対し、一点集中型であるため、SessionManager が落ちた場合、機能しなくなることである。

3. リモートエディタの概要

リモートエディタの通信は図 1 に示す様に、スター型の構成となる。中央の SessionManager はデーモン型のサーバで、Session と呼ばれる単位でテキストの編集を管理する。エディタと SessionManager 間の通信は REP パケットを用いて行われる。

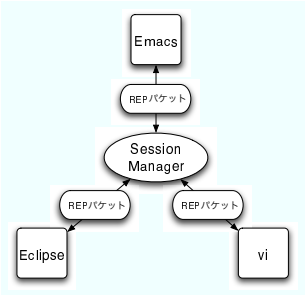

図 1 SessionManager と各エディタの構成

各エディタはローカルのセションマネージャに接続し、図 2 のようにスター型を構成し、各エディタで行われた編集はセションマネージャに送信され、セションマネージャから各エディタへと送信される。図 2 ではリモートエディタと SessionManager の接続自体はスター型となっているが、パケットは図の矢印で示すように、各エディタをリレーの様に送信され、擬似的なリングネットワーク構成となっている。

また、異なるホスト上にあるセションマネージャどうしが図 3 のように接続され、異なるマシン上にあるリモートエディタどうしのテキスト編集を可能にしている。

以下に、REP のパケットと REP コマンドについて示し、リモートエディタと SessionManager との接続からテキスト編集開始までの流れを説明し、Merge について説明する。

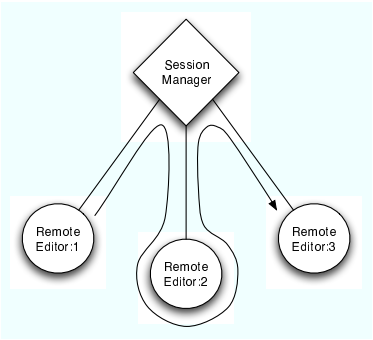

図 2 ローカルのネットワーク構成

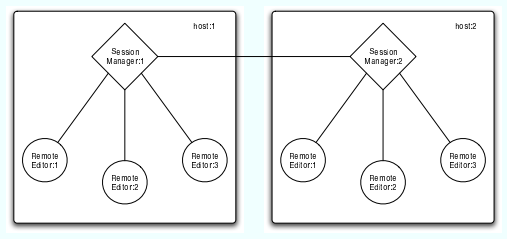

図 3 SessionManager どうしの接続

3.1 REP のパケット

REP で用いるパケットの構成は図 4 のようになる。パケットのはじめの 24Byte はヘッダ部分で、コマンドの種類や EditorID、SessionID などの情報が記述されている。後ろの可変部分はテキスト情報が記述されている。ヘッダ部分は cmd,sid,eid,seqid,lineno,textsize となっており、以下に示す情報を保持する。

- cmd：コマンドの種類を示す。これにより、各エディタはこのパケットを受信したときの動作を決定する。
- sid：SessionID を示す。エディタが現在参加している Session の番号。
- eid：EditorID を示す。エディタに割り振られた番号。
- seqid：SequenceID。コマンドの識別番号。
- lineno：編集されたテキストが、そのファイルの何行目にあるかを示す。
- textsize：実際に編集されたテキストのサイズ。

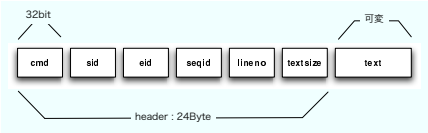

図 4 REP のパケット構成

3.2 REP コマンド

REP コマンドはテキスト編集における基本的な操作を表している。エディタごとの編集コマンドをこの REP コマンドへ変換することにより、各エディタ間の編集コマンドの差を埋めることができる。REP では以下に挙げる編集コマンドが提供されている。これらのコマンドを用いてアプリケーション間の協調動作を実現する。いくつか例を挙げる

- read:
  バッファにテキストを読み込むためのコマンド。引数に行番号を与える。このコマンドを受け取ったアプリケーションは行番号に対応するテキストを insert コマンドを用いて送信元のアプリケーションに返す。
- insert:
  バッファに新たに挿入された行を、接続されている別のアプリケーションのバッファに反映させるためのコマンド。引数に行番号とテキストを与える。このコマンドを受け取ったアプリケーションは、行番号に対応するテキストを自分が持つバッファに挿入する。
- delete:
  バッファで削除された行を、接続されている別のアプリケーションのバッファに反映させるためのコマンド。引数に行番号を与える。このコマンドを受け取ったアプリケーションは、行番号に対応する行を削除する。
- replace:
  バッファに既存の行に対して行われた編集を、接続されている別のアプリケーションのバッファに反映させるためのコマンド。引数に行番号とテキストを与える。このコマンドを受け取ったアプリケーションは、行番号に対応する自分が持つバッファに挿入する。テキストを自分が持つバッファの行番号に対応するテキストを置換する。

3.3 SessionManager への接続

実際の SessionManager への接続の流れを説明する。図 5
リモートエディタを起動すると、デフォルト (ローカルホスト) の SessionManager へと接続される。接続が確立すると、リモートエディタは SessionManager に対し、join コマンドを送信する。join コマンドを受け取った SessionManager はリモートエディタに対し Session リストを送信する。リモートエディタは受け取った Session リストをユーザへ示し、ユーザは適当な Session を選択する。ユーザの選択行為により、リモートエディタは SessionManager へ select コマンドを送信し、SessionManager がわでは、このエディタが選択された Session に追加される。その後テキスト編集が開始される。

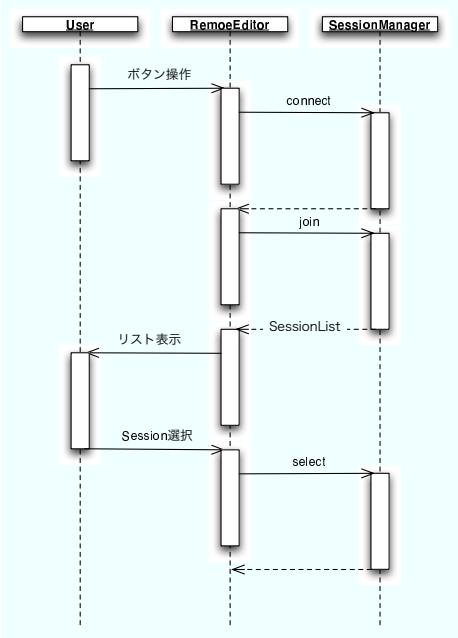

図 5 SessionManager への接続の流れ

3.3.1 put コマンド

put コマンドは、現在エディタで開いているドキュメントを Session として追加するコマンドである。前述と同じ様に join コマンドを送信した後のタイミングで、リモートエディタは put コマンドを送信し、put された Session を select する。SessionManager では filename を受け取り、Session を生成し、Session リストに追加する。

3.4 マ ー ジ

複数人で編集を行う場合、各ユーザが勝手に編集を行っても最後には各バッファが等しくなければならない。これを保証するには、独立して行われた各リモートエディタの編集を一つの編集シーケンスにマージする必要がある。

4. Emacs, vim 版リモートエディタ

4.1 vi 版のリモートエディタ

vim にはキーボード入力を監視しているループがあり、vim 版のリモートエディタでは、REP で用いるソケットをこのループ内の select() で監視し、キーボード入力と REP パケットの受信の処理を同一のループ

内で行いっている。

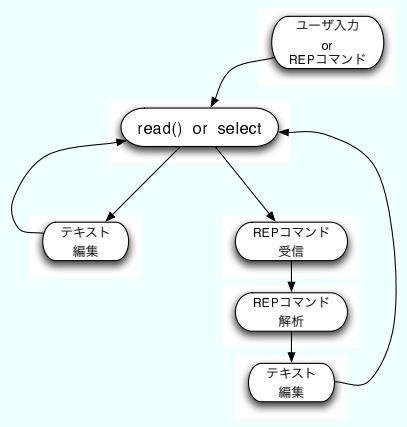

図 6 vim でのテキスト編集

4.2 Emacs 版のリモートエディタ

Emacs では Emacs-Lisp を用いて実装されている。Emacs には様々なイベントを処理する変数としてフックが存在する。Emacs 上の実装では次のようなフックを用いている。

- window-scroll-functions
  Emacs 画面が更新 (再描画) される時に呼び出されるフック

- before-change-function
  現在のバッファに変更が加えられる直前に実行される

- after-change-functions
  現在のバッファに変更が加えられた直後に実行される

また、SessionManager とエディタ間の通信には、Emacs とは別に、コマンド受信プログラムを作成し、そしてそれを Emacs 内部で実行することで Emacs-Lisp とプロセス間通信により、Emacs 本体と受信プログラムとの通信を行っている。

そして、Emacs-Lisp の編集コマンドを用いて受信プログラムから送られて来た処理をバッファに対して実行している。

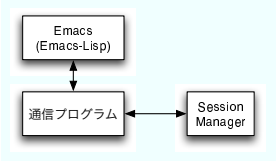

図 7 Emacs での通信

5. SessionManager の実装

5.1 SessionList

SessionManager は SessionList オブジェクトを持っており、このオブジェクトにより、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、ファイル名を管理する。SessionList オブジェクトは Session オブジェクトの HashTable を持っており、またそれぞれの Session オブジェクトは Editor オブジェクトのリストを持っている。図 8

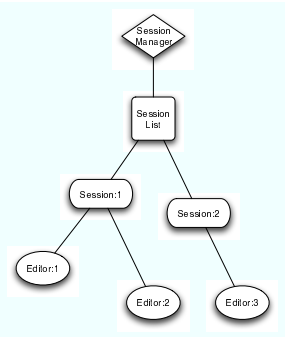

図 8 SessionList オブジェクト

SessionManager はリモートエディタから PUT コマンドを受信すると、図 8 に示すように、SessionList オブジェクトに Session オブジェクトが追加される。Session オブジェクトには、master となるエディタのソケット情報やファイル名などの情報を保存する。その後 SELECT コマンドを受信すると Session オブジェクトに対し addEditor メソッドで Editor オブジェクトを追加する。Editor オブジェクトはエディタのソケット情報やエディタ ID などの情報を保持する。図 9

図 9 SessionList オブジェクト

5.2 SessionManager どうしの接続

SessionManager どうしが接続されていない状態では、各 SessionManager ではそれぞれ独自に SessionID を割り振ることになる。その後、SessionManager どうしが接続すると、SessionID が競合する可能性がある。これを解決するため、変換テーブルを作成し、各 SessionManager では独自に割り振った Ses-

sionID を変更することなく、SessionID を変換し、他の SessionManager へ編集コマンドなどを送信を行う実装とする。
SessionManager どうしが接続されると、接続先に存在する Session のリストを自身の Session リストに追加し、その際、SessionID を競合しないように振り直し、同時に変換テーブルを生成する。(図 10)

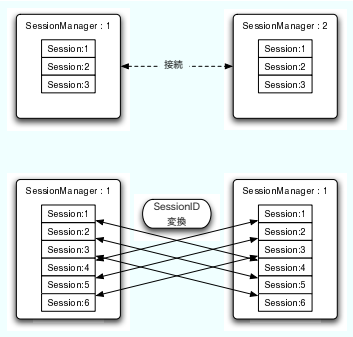

図 10 SessionID の変換

6. Eclipse

Eclipse はオープンソースの統合開発環境 (IDE) の一つであり、Java により記述されている。さらに、Eclipse はプラグインとして様々な機能を組み込むことができるよう設計されていて、拡張性が高くできている。今回はリモートエディタをこの Eclipse のプラグインとして実装する。

6.1 RemoteEditor クラス

リモートエディタを Eclipse へ実装するため、まず、プラグインを作成し、エディタを定義する。Eclipse のプラグインとしてエディタを定義するには”org.eclipse.ui.editors”拡張ポイントを使用する。

ここでは、エディタとして使用するクラスとして、”RemoteEditor”クラスを指定している。
Eclipse でプラグインとして新たにエディタを追加するため、Eclipse 既存の TextEditor クラスを拡張した RemoteEditor クラスを作成する。RemoteEditor クラスには、ITextListner と REPCommandListener という二つのインターフェースが implements されていて、それぞれ、テキストの編集を処理する機能と、コマンドの受信を処理する機能となっている。

6.2 ユーザインターフェース

SessionManager 接続時のユーザ操作においては、Eclipse 既存の拡張ポイントである、actionSets を利用する。これにより、ツールバー上にボタンが追加され、ユーザはボタン操作によって、SessionManager に接続する。

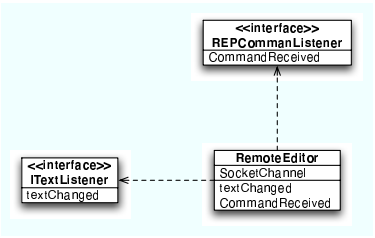

図 11 RemoteEditor クラス

6.2.1 ボタンの処理

ここでボタン操作を定義するためのクラスとして、RemoteEditorAction クラスを定義している。
ボタンが押されたときの処理を RemoteEditorAction の run メソッド内に記述する。

表 1 RemoteEditorAction

run	アクション (ボタン操作) が起きた時の操作を記述するメソッド

Workbench オブジェクトからアクティブなエディタを取得し、現在開かれているファイルの EditorInput オブジェクトを取得する。そして、RemoteEditor クラスを用いて取得したファイルを開く。

6.3 テキスト入力

ユーザからのキーボード入力は、Eclipse のメインループが監視しており、RemoteEditor クラスに ITextListener インターフェースを implements することにより、ユーザからテキストへの編集があったときに、RemoteEditor クラスの textChanged メソッドが呼び出される。

この textChanged メソッドはユーザからのテキスト編集があると呼ばれるようになっている。ここでメソッドの引数となっている TextEvent オブジェクトからは、offset や編集されたテキストなど、REP コマンド生成に必要な情報が取得できる。このメソッド内ではテキスト全体の行数の変化により、コマンドの種類を決定し、REPCommand オブジェクトを生成している。textChanged メソッドが呼ばれた時に、テキスト全体の行数が増えている場合は insert コマンド、行数が減っている場合は delete コマンド、行数が変化しない場合は replace コマンドを発行する。

行数増 → insert
行数減 → delete
行数同 → replace

6.4 コマンド受信

RemoteEditor クラスは CommandReceived メソッドをもつ REPCommandListener インターフェース

が implements されており、SessionManager からコマンドを受信すると、この REPCommandReceived メソッドが呼び出される。
RemoteEditor クラスで CommandReceived メソッドによりコマンドを受信し、コマンドを解析する。そして、解析されたコマンドを switch 文により処理する。

6.5 ネットワーク部分

ネットワーク部分は、送信部分の REPSender クラスと受信部分の REPReceiver クラスに機能わけされている。

6.5.1 送信部分

送信部分では RemoteEditor により生成された REPCommand オブジェクトを REP のパケットに変換し、SessionManager へ送信する。

表 2 REPCommandSender

pack	REPCommand オブジェクトから REP のパケットを生成するメソッド
send	生成されたパケットを送信するメソッド

REPCommand オブジェクトから REP のパケットを生成するには、REPPacketSender クラスの pack メソッドを使用する。pack メソッドは REPCommand オブジェクトを受け取ると、ByteBuffer クラスの putInt メソッド、putChar メソッドにより ByteBuffer オブジェクトを生成する。
send メソッドは生成された ByteBuffer オブジェクトを SessionManager へ送信する。

6.5.2 受信部分

パケット受信部分は単独のスレッドになっており、コマンドが受信されると、REPCommandListener が implements された RemoteEditor クラスの CommandReceived メソッドが呼び出されるようになっている。
受信部分では、受信した REP のパケットを REPCommand オブジェクトに変換する。

表 3 REPCommandReceiver

unpack	パケットを受信し、REPCommand オブジェクトを生成するメソッド
run	implement された Runnable インターフェースの run メソッド。 パケットを非同期に受け取るスレッドとなる。

受信スレッドにより SessionManager からのパケットを受信し、unpack メソッドにより、REPCommand オブジェクトを生成する。run メソッド内で呼び出している CommandReceived メソッドは REPCommandListener インターフェースが implements された RemoteEditor オブジェクトであり、REPCommandEvent を渡している。

6.6 REPCommand オブジェクト

Eclipse リモートエディタ内での REP コマンドの様々な処理は、REPCommando オブジェクトを用いて行う。REPCommand オブジェクトの生成には 2 つの種類がある。
まず 1 つは、SessionManager から受信した REP のパケットを REPCommand オブジェクトに変換するもの (6.5.1) で、もう 1 つは、ユーザからのテキスト入力を REPCommand オブジェクトに変換するものである (6.3)。図 12

図 12 REPCommand オブジェクトの生成

6.7 ドキュメントへの反映

REP は独自のスレッドを用いて REP コマンドを受け取っている (図 13)。しかし、Eclipse では GUI のメインスレッド以外からドキュメントへの編集を行おうとすると、SWTException が発生する。このような制限を行うのは、GUI に対し、複数のスレッドから非同期にアクセスすると GUI の整合性が失われる可能性があるためである (図 14)。これを防ぐには、GUI のイベントディスパッチスレッドである Display オブジェクトの syncExec メソッドを経由して、GUI のメインスレッドに編集処理を実行させる必要がある。

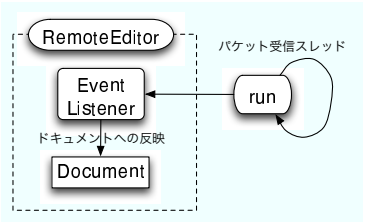

図 13 REP パケット受信スレッド

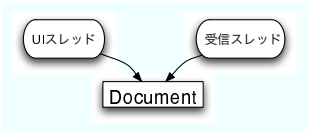

図 14 テキスト編集の競合

パケット受信スレッドにより、パケットが受信されると、REPPacketReceived クラスの unpack メソッドにより REPCommand オブジェクトが生成される。REPCommand オブジェクトを受け取った RemoteEditor オブジェクトは changeText メソッドにより REPCommand オブジェクトを Eclipse のテキスト編

集コマンドへ変換する。この編集処理を anonymous class として syncExec メソッドに渡すと、Display オブジェクトのイベントキューに追加される。その後 GUI スレッドにより、キューに追加された処理が適当なタイミングで実行されドキュメントへの反映が行われる。この流れを図 15 に示す。

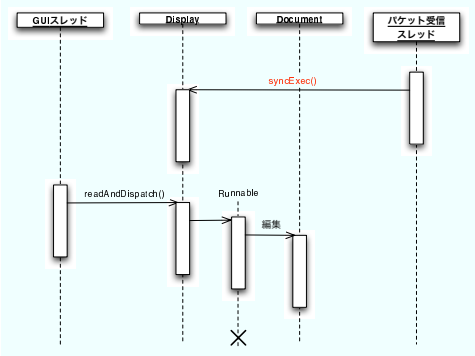

図 15 ドキュメントへの反映

7. まとめと今後の課題

7.1 ま と め

今回はリモートエディタを Eclipse のプラグインとして実装することにより、Eclipse と Emacs、vi を相互に接続することを可能とした。また、今回作成した、REP コマンドを生成し、送信するための REPCommandSender クラスと、SessionManager からパケットを受信し、REPCommand オブジェクトを生成するための REPCommandReceiver クラスの 2 つのクラスを使用することにより、他の Java アプリケーションへの REP の実装がより簡単に実現できる様になることが予想される。

そして、SessionManager においては、仕様を変更し、リモートエディタからの接続手続きを簡略化することにより、ユーザの負担を軽くすることができた。

7.2 今後の課題

今後は REP のエディタへの実装だけではなく、MindMap などのエディタ以外への応用を行うことも課題となっている。REP を実装する MindMap アプリケーションとして、Java アプリケーションである FreeMind などがある。

参 考 文 献

1) 双方向リモートエディタの vim への実装, 安村恭一, 河野真治,2004 年 2 月
2) 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 日本ソフトウェア科学会第 17 回論文集、2000
3) 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 宮里忍、河野真治 日本ソフトウェア科学会第 20 回論文集、2003
4) 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
5) 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
6) Building Real Time Groupware with GroupKit, A Groupware Toolkit ,1996
7) SOBA Project http://www.soba-project.org/jp/