2009－GN－71 (9)
2009／3／18

ソフトウェア開発における協調作業のための アプリケーション間通信プロトコル

宮城健太† 河野真治††

我々が提案しているRemote Editing Protocol(REP)は、テキスト編集に特化したアプリケーション間通信プロトコルである。ここでは、エディタへのユーザ操作の煩雑さを改善するため、REPの接続プロトコルの変更を行った。また、マージアルゴリズムについても変更を行い、その変更に伴う問題点の解決法についても論じる。

InterApplication Communication Protocol for Cooperative Work in Software Development

KENTA MIYAGI † and SHINJI KONO††

Remote Editing Protocol which we introduced is inter-application communication protocol for text editing. In this paper we improve the connecting protocol to remove cumbersome operation of user interface of Remote Editor. In addition, we discuss about solution of the problem that caused by changing the Merge Argorithm.

1. はじめに

我々の研究室で提案しているRemote Editing Protocol(REP)は、テキスト編集に特化したアプリケーション間通信プロトコルである。REPを汎用のテキストエディタへ実装し、そのエディタ同士を接続することで、相互にデータの編集作業を行うことが可能になる。

ソフトウェア開発において、複数人での協調作業を行う際、問題となるのが、距離的な制約や、アプリケーションの違いや、キーボードの違い(USキーボードと日本語キーボードの差など)の様な環境の違いであると考える。また、同じアプリケーション同士であっても、カスタマイズの方法によって、操作方法が大きく異なることもある。

REPはこのような距離的な制約や、環境の違いなどを吸収し、協調作業の効率化を図ることにより、ソフトウェア開発の生産性、信頼性の向上を目指している。

以前のREPは[2)~11)]エディタ同士を直接接続し、データの相互編集を行っていた。その場合、エディタ側から相手のIPアドレスやファイル名の入力などの操作を行わなければならなかったため、ユーザ操作が煩雑であった。そこでREPでは、エディタの接続や、テキスト編集(Session)を管理するためのSession Managerの導入を行った[1)]。しかし、この場合でも、リモートホスト上にある他のエディタとの編集作業を行う際にはそのIPアドレスを入力しなければならない。

本研究では、Session Mnager同士の接続を導入した。Session Managerはそれぞれのコンピュータに1つずつ存在し、エディタはデフォルトのSession Managerへの接続操作を行うだけよくなった。リモートホストとの相互の編集を行う際にはSession Manager同士を接続し、Session Managerを介して編集を行うプロトコルに変更し、実装した。

REPではエディタ同士の編集の衝突を解決するために、エディタ間でのデータの不整合を解消するマージの導入を行っている。以前までのマージアルゴリズムは、1対1の通信に対応したアルゴリズムであったため、これを複数人で同時に編集を行うアルゴリズムに変更した。また、以前のプロトコルではマージの処理をエディタ側で行っていたが、マージの処理はREPのプロトコルにおいて共通の機能であるため、Session Manager側へ実装することが望ましいと考え、Session Managerへ移行した。

† 琉球大学理工学研究科
Graduate School of Engineering and Science, University of the Ryukyus
†† 琉球大学工学部情報工学科
Information Engineering, University of the Ryukyus

しかし、マージの処理を Session Manager 上で行なうために、マージコマンドとエディタコマンドとの間に衝突が起こる可能性がある。マージの処理をしている最中にユーザが入力する可能性があるためである。この問題点の解決方法についても考察する。

2. Remote Editing Protocol(REP)

2.1 REP の概要

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同士を接続し、データの直接編集を相互に行うことを目的としたプロトコルである。REP を実装したテキストエディタをリモートエディタと呼ぶ。現在、REP は Vim, Emacs, Eclipse などに実装されている。REP では REP コマンドを用いて、エディタの接続の処理や、データの相互編集を行う (図 1)。

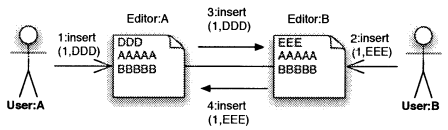

図 1 REP での相互編集

2.2 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。REP は、様々なアプリケーションに実装されているため、それぞれのエディタ間での編集コマンドの差を埋める必要がある。REP では、それぞれのエディタの複雑な編集コマンドを REP コマンドの insert、delete コマンドに変換してデータの相互編集を行っている。

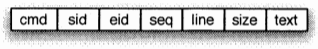

図 2 REP コマンド

以下に、REP コマンドのそれぞれのフィールドについての説明を行う。

- cmd
  コマンド識別子。REP ではこの識別子によって処理の内容を判断する。
- sid
  Session を識別するための番号。
- eid
  エディタを識別するための番号。
- seq
  シーケンス番号。
- lineno
  編集を行う行番号。
- text
  編集された/するテキスト。

次に、テキスト編集に用いられる REP コマンドと、接続時に用いられる REP コマンドの一部を紹介する。

編集コマンド

- insert
  テキストへの行の挿入を表すコマンド。エディタでテキストへの行の挿入があった場合、このコマンドを発行し、たのエディタへ送信する。lineno に行番号、text に挿入するされたテキストをセットする。受信したエディタは指定された行にテキストを挿入する。
- delete
  テキストからの行の削除を表すコマンド。エディタで行が削除された時に発行され、text には削除された行がセットされる。このコマンドを受け取った側のエディタでは、テキストに対し、行の削除を行い、このコマンドの text の値を、このエディタで削除した行へとセットし直す。

接続コマンド

- join
  エディタが Session Manager に接続するときに発行するコマンド。このコマンドの場合、エディタがどの Session に接続するかはすぐには決定せず、ユーザの Session Manager への GUI 操作によって決定する。
- put
  エディタが Session Manager に接続し、Session を生成するコマンド。この場合。エディタは生成された。Session へ自動的に接続する。このとき生成される Session はエディタ側が現在開いているテキストとなる。
- sync
  ある Session に参加する全てのエディタに対してテキスト内容の同期を行う。このコマンドは Session に新たにエディタが接続してきたときに Session Manager から file owner のエディタへ送信される。受け取ったエディタは自分が持っているテキ

ストの内容を insert/delete コマンドにより他のエディタへ送信して、テキストの内容を反映させる。

2.3 Session Manager の導入

最初の REP は、一つのエディタをサーバとして、クライアントからの編集を可能にするものであり、1997年に新垣将史によって、pico エディタに実装された[10]。その後、Emacs 上にも実装された[8]。この版の REP では、変更は一方向であり、サーバはクライアントからの変更を受け付けるだけであった。

宮里忍、安村恭一は、二つのエディタ間で REP コマンドを巡回させることにより、二つのエディタ間での双方向の編集を可能にするマージアルゴリズムを作成した[2)4)]。この時に、REP は vim にも移植されている。

2007 年には Eclipse への実装を行なった[1)]。これにより、Eclipse 同士、あるいは、Eclipse 上のファイルを Emacs/vim などで編集することが可能になる。

この時点で、エディタ同士を直接接続する手順が複雑すぎることが明らかになった。エディタの IP アドレスやファイル名をユーザがエディタから直接入力するのは、かなり繁雑である。そこで、エディタ間の接続を管理する GUI として、Session Manager を導入することを提案した (図 3)。

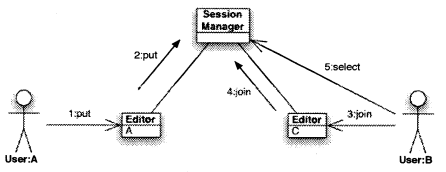

図 3 Session Manager の導入

Session Manager にはネットワーク経由でどこからでも接続できるが、その場合は、やはり、エディタの方で IP アドレスなどを指定する必要がある。それよりは Session Manager はコンピュータ上に一つあると言う方が望ましい。そうすれば、エディタは決まったポートでデフォルトの Session Manager に接続できる。

このように Session Manager を導入することで、エディタ側に複雑なユーザインタフェースを実装することなく REP を実装することが可能となる。このユーザインタフェース部分は、Emacs では 10%程度、vim では 30%程度を占めており無視できない大きさである。

2.4 Session Manager 同士の接続

リモートホスト同士のエディタ間で相互にデータの

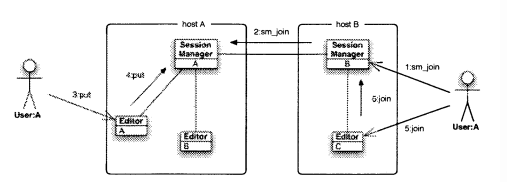

図 4 Session Manager の同士の接続

編集作業を行うにはまず、そのホストの Session Manager 同士を接続し、エディタ間での通信は接続された Session Manager を介して行う。

Session Manager 同士を接続するにはユーザが GUI 操作によって接続先の Session Manager を決定し、sm_join コマンドを発行する。sm_join コマンドを発行して接続してきたほうを slave、接続されたほうを master とする。

sm_join コマンドを受け取った Session Manager が既に master だった場合、sm_join_ack を全ての slave に送信する。slave だった場合は master の方向へ sm_join コマンドを送信する。slave が ack を受け取った場合は Tree 構造の子の方向へ ack を送信する。

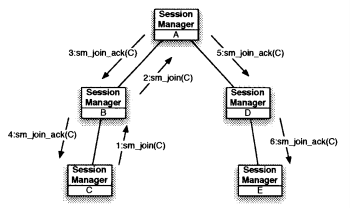

図 5 sm_join コマンド

REP では、Session Manager ID(smid) をユニークに保つため、Session Manager 同士の接続は Tree 構造になるように制限を行っている。そこで Session Manager 同士の接続をリング状にさせないため、ループの検出を行う必要がある。これ以外にも複数の Session Manager に接続させない、Session を持っている Session Manager は接続できない、などの制限が必要になる。

2.5 エディタの接続

REP を使用して複数のエディタ同士でテキストの相互編集を行うにはまず、エディタへのユーザの操作により join コマンド、または put コマンドを発行する。すると、エディタは自動的にデフォルトの Session Manager に接続する。put によって接続すると Session Manager 側に Session が生成される、エディタは自動的にその Session に接続する。join で接続を行う

と、このエディタ接続が Session Manager 側の GUI に反映される。そして、ユーザは Session Manager への GUI 操作により、join によって接続したエディタがどの Session に接続するかを決定する。

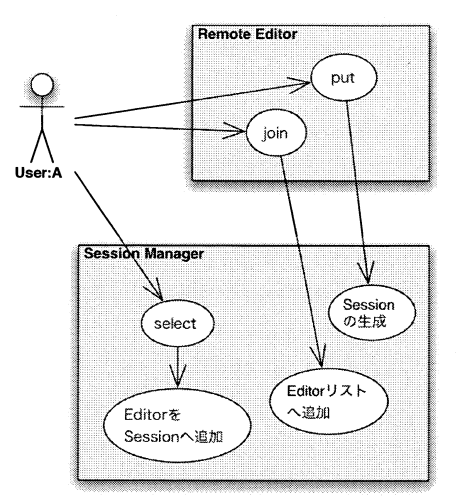

図 6　エディタの接続

リモートホスト間で通信を行う場合にはまず、Session Manager どうしの接続を行う。この接続は Session Manager へのユーザ操作によって行う。

Session Manager 同士の接続が確立したら、エディタ側から put/join コマンドによりローカルホストの Session Manager へ接続を行う。これを受けとった Session Manager は、Session の追加やエディタの追加を master の Session Manager に join/put を経由し、変更を知らせる。master から全ての slave に対しては join_ack、put_ack によって知らせる。このとき、同時

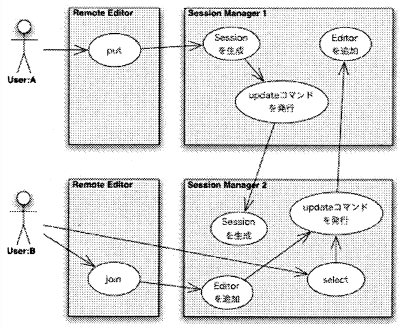

図 7　Session Manager を介したエディタの接続

Session Manager をまたいだエディタの接続を行う場合、Session ID(sid) と、Editor ID(eid) は smid を含んだ値になる。smid が全ての Session Manager の間でユニークであるため、sid、eid、も全ての接続の間でユニークな値となる。

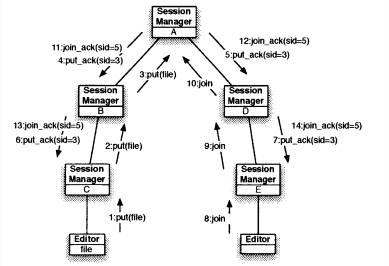

図 8　join/put の動作

2.6　select コマンド

join コマンドにより Session Manager へ接続したエディタはその時点ではまだ接続する Session が決定していない。ローカルホスト内だけに閉じた状態であれば図 6 で示したように、Session Manager 側の GUI 操作によりエディタを Session へ追加するだけで良い。

通常、エディタが接続したい Session は Session Manager の接続先であるリモートホストにあるので、Session Manager を介した操作が必要になる。これは select コマンドにより行う。また、他の Session Manager へ、エディタが新たに Sessin に接続したことを通知する必要がある。これは select_ack コマンドにより行う。

select コマンドは file owner がいる Session Manager へ送信される。file owner の方向は put/join、join_ack、put_ack によって生成されたルーティングテーブルによって知ることができる。そこで join したエディタが Session へ追加される。そのときエディタの Sessiono への追加を他の Session Manager にも知らせるために update コマンドを発行する。update コマンドは master の Session Manager へと送信され。master が発行する update_ack コマンドによって全てのエディタへと通知される (図 9)。

3.　マ　ー　ジ

REP では、それぞれのエディタでのテキストへの編集を非同期に行うため、エディタ間で編集コマンドの衝突が起こることがある。

例えば、エディタ A とエディタ B でほぼ同時に同

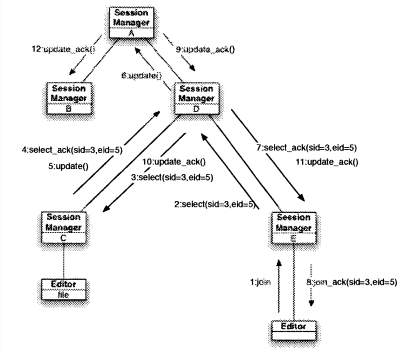

図 9　select コマンド

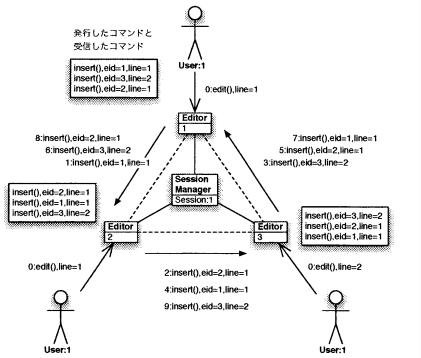

図 11　Session Ring 上の REP コマンドの送信

じテキストの同じ行を編集した場合、2つのエディタ間で編集の順番が逆になってしまい、編集結果に差が出てしまう (図 10)。

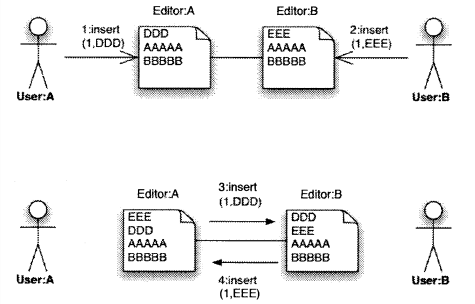

図 10　コマンドの衝突

このような編集コマンドの衝突による編集結果の不整合をマージにより修正する。

3.1 Session Ring

実際にはエディタは Session Manager にスター型に接続されるが、同じ Session 内 (同じテキストに対する編集を行っているエディタ同士) での REP コマンドの送受信は仮想的なリングネットワークを構成し、このリング上を REP コマンドを 1 方向に送信する (図 11)。これを Session Ring と呼ぶ。テキスト編集を REP コマンドに変換し、この REP コマンドをリングネットワーク上に巡回させることにより、全てのエディタで起こった編集を REP コマンド列として、それぞれのエディタで収集する。この Session Ring を用いると、全てのエディタで起こったテキスト編集の履歴を REP コマンド列としてそれぞれのエディタで蓄積できる。この蓄積されたコマンド列を利用してマージを行う。

3.2 マージアルゴリズム

それぞれのエディタに蓄積された REP コマンド列は順番が異なる。図 11 を例に見ると、Editor1 では eid=1,3,2 の順に蓄積されているが、Editor2 では eid=2,1,3 の順番である。これをソートして順番を一致させてエディタへ反映すれば編集結果を一致させることができる。しかしエディタには既に、それぞれのエディタ間で異なった順番で REP コマンドが反映されてしまっているので、エディタ側のテキストに対して Undo の処理を行う。これにより、それぞれのエディタは、編集前の状態に戻り、テキストは一致している状態である。その後、ソートされた REP コマンド列をテキストへ反映させ全てのテキストを一致させる。

3.3 マージ処理の Session Manager への移行

マージの処理はこれまで、エディタ側で実装されていたが、マージは REP 共通の処理であるため、それぞれのテキストエディタへ個別に実装するよりは、Session Manager へ実装する方が望ましいと考えた。しかし、エディタと Session Manager との間では非同期に通信を行っているため、マージコマンドとエディタコマンドとの間で衝突が起こってしまうことがある (図 12)。マージ中にエディタからの割り込みがあった場合である。これを解決するため、マージコマンドとエディタコマンドとの間マージを行うリマージコマンドの生成を提案した。

4. 他の研究との比較

4.1 GroupKit

GroupKit とはグループウェアドローツールやエディタ、遠隔会議システムなどの、リアルタイム分散アプリケーションの開発のためのツールキットであり、Tcl/Tk のライブラリとして提供されている。

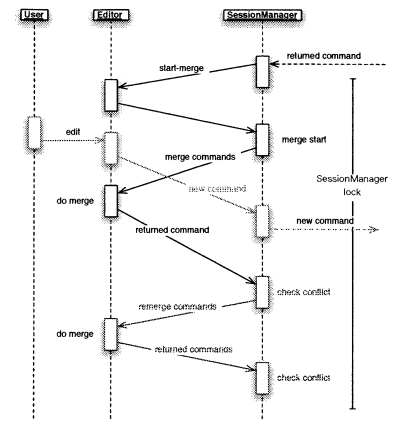

図 12　リマージ

REP と比較すると、GroupKit は Tcl/TK により記述されているため、Emacs や vi などの汎用なエディタに実装することができない。また、サーバ 1 つに対し複数の Session Manager が接続するプロトコルとなっているため、ネットワークの負荷や処理の負荷が一点に集中してしまう仕組みになっている。これに対して REP はサーバは分散型である。

4.2 SOBA プロジェクト

SOBA プロジェクトは京都大学を中心とする産学官共同によるグループウェアの研究プロジェクトである。SOBA とは、Session Oriented Broadband Applications の略で、複数のユーザが多様なメディア (映像、音声、アプリ画面やテキストなどのデータ) 情報を共有、享受することができる P2P 型ネットワークアプリケーションである[12]。

SOBA プロジェクトがフレームワークを提供しているのに対し、REP はプロトコルを提案している。REP がプロトコルであることの利点として、様々な既存のアプリケーションに対して REP 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が可能であり、リモートエディタを使用するユーザの慣れ親しんだ環境でテキストの編集作業を行うことができる。逆に REP では既存のアプリケーションへの実装を目的としているため、そのアプリケーションの膨大なソースコード解析し理解しなければ実装できないため、SOBA フレームワークを用いたアプリケーションの開発に比べて難しくなっている。その他の欠点として、SOBA が P2P ネットワーク通信によるアプリケーションであるのに対して、REP はサーバ、クライアント型方式である。そのため、SOBA に比べて Session Manager に負荷が集中しやすく、また、ある Session を保持している Session Manager に不具合があると、その Session Manager に接続しているエディタ全てが影響を受けてしまう。

5. ま と め

本研究では、REP に Session Manager 同士の接続を行うプロトコルを追加した、これによって、ユーザからのエディタに対する IP アドレスの入力処理などの煩雑な操作をなくすことができた。

また、以前のマージアルゴリズムは 1 対 1 の通信に対応したアルゴリズムだったため、新たに多対多の通信に対応したマージアルゴリズムを提案した。

この、マージの処理は REP のプロトコルにおいて共通の機能であるため、Session Manager 側へ実装することが望ましい。しかし、マージの処理を Session Manager 上で行なうと、マージコマンドとエディタコマンドとの間に衝突が起こる可能性がある。マージの処理をしている最中にユーザが入力する可能性があるためである。この問題を解決するために、リマージコマンドの生成のアルゴリズムを提案し、実装した。

参 考 文 献

1) 宮城健太, 河野真治. “リモートエディタの Eclipse への実装”. 2007.
2) 安村恭一, 河野真治. “双方向リモートエディタの vim への実装”. 2004.
3) 安村 恭一, 河野 真治. “巡回トークンを用いた複数人テキスト編集とセッション管理”. June, 2004.
4) 宮里 忍, 河野 真治. “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”. Sep, 2003.
5) 宮里 忍, 河野 真治. “リモートエディタの SVG への応用”. Sep, 2002.
6) 宮里 忍, 河野 真治. “リモートエディタの日本語ターミナルへの応用”. June, 2002.
7) 河野 真治, 宮里 忍. “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”. July, 2001.
8) 新垣将史, 河野真治. “Remote Editor on Emacs”. May, 2000.
9) 新垣将史, 河野真治. “Remote Editing Protocol を用いた複数ユーザ編集システム”. September, 2000.
10) 新垣将史, 河野真治. “リモートエディタの実装と、その XML への応用”. September, 1999.
11) 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有効性”. May, 1999.
12) “SOBA Project”, http://www.soba-project.com/.
13) Mark Roseman and Saul Greenberg. “Building Real Time Groupware with GroupKit, A Groupware Toolkit”. 1996.