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

安 村 恭 一[†]　河 野 真 治[☆]

我々は複数人によるテキスト編集を可能にするリモートエディタを提案、実装してきた。リモートエディタは Remote Editing Protocol(REP) というコマンドベースのプロトコルを用いて相互編集の整合性を保っている。ここでは、REP コマンド列をトークンとし、リング状に接続した各リモートエディタ間を巡回させることによる複数人テキスト編集について述べる。また、テキスト編集のセッション管理やリモートエディタ間の接続をサポートするセッションマネージャについても説明する。

Multi-user text editing and session management using Circulating Token

Kyoichi Yasumura[†] and Shinji Kono[☆]

We propose and implement Remote Editor which enables multi-user text editing. Remote Editor keeps correctness of mutual edits using Remote Editing Protocol (REP), a command based protocol. This paper describes multi-user text editing using Circulating Token, which is created by REP commands and circulates on each Remote Editor. Then, we explain Session Manager. Session Manager supports connection for each Remote Editor and manages a session of multi-user text editing.

1. はじめに

複数人のユーザが同時にひとつのファイルへ編集を行う際、編集箇所の競合が発生する場合がある。そのような競合は CVS などを用いて編集作業後に解消させるか、複数人が同時に編集することができないようにする必要がある。

我々はこれまで、異なるマシン上のテキストエディタを直接接続し、テキスト編集に特化したプロトコルである Remote Editing Protocol (以下 REP) を用いてテキスト編集を行うリモートエディタを開発してきた[1)5)]。REP はテキスト編集時における一般的な編集コマンドを統一したプロトコルである。リモートエディタで編集を行うと、その編集を行単位で逐次 REP の編集コマンドに変換する。この変換された REP コマンドの列を用いて他のリモートエディタへの編集を行う。この REP コマンド列を保持しておくことで非同期な編集を行うことができ、ネットワークの遅延・切断に強い遠隔テキスト編集を可能にしている。

また、REP によって統一された編集コマンド群を用いることにより、エディタ間の編集コマンドの差を埋めることができるので、ユーザは好きなエディタを使うことができる。

リモートエディタはこれまで pico や Emacs, vim, TextEdit などに実装され、その有効性を示してきた[2)3)4)]。本稿では、REP を巡回トークンを用いて配信することによるリモートエディタ間のテキストの整合性の保証について説明する。また、セッションへの参加やエディタ間の接続のサポートをするセッションマネージャについても述べる。

2. リモートエディタ

リモートエディタは REP を用いて異なるマシン上の REP が実装されたアプリケーションの保持するバッファを開いて編集することができる。複数人による編集では、一方がバッファを編集するともう一方のバッファにも変更が反映されて、お互いのバッファを同時に編集しあうことができる。前者を単方向編集と、後者を双方向編集と呼ぶ。

ここでは REP について説明をし、REP を用いた単方向と双方向の編集について述べる。

2.1 Remote Editing Protocol

REP コマンドはテキスト編集における一般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 (図 1)。

† 琉球大学理工学研究科情報工学専攻
Interdisciplinary Information Engineering, 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.

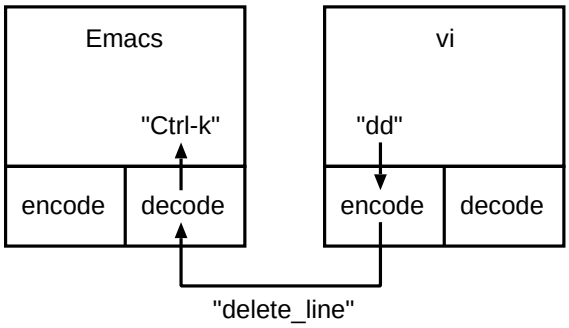

図 1 エディタ間の編集コマンドと REP コマンド

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。

- open:
  ファイルを開き、バッファリングを行うコマンド。引数としてファイル名を与える。このコマンドを受け取るとアプリケーションはファイルを開いてバッファへ展開する。
- 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。
- save:
  バッファをファイルへ書き出すコマンド。このコマンドを受け取ると、テキストバッファをファイルへ書き出す。
- close:
  編集を終了するコマンド。このコマンドを受け取ったアプリケーションは、コマンドを送信したアプリケーションとの接続を切断する。
- 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。

実際にバッファへの編集を行うコマンドは insert と delete, replace だけである。これらのコマンドは行単位の編集のみを行う。つまり、ある行の 1 文字を編集すると、その行自体を置換することになる。

2.2 単方向編集

単方向のファイル編集は、一般のエディタの機能をファイル管理と編集部分に分け、それぞれがサーバとクライアントのような型になっている。図 2 では host A がファイル管理を行うサーバで、 host B がテキスト編集を行うクライアントである。ユーザは host A のマシン上にあるファイルを REP を用いて接続したリモートエディタのクライアントプログラム (Remote Editor B) で編集することができる。

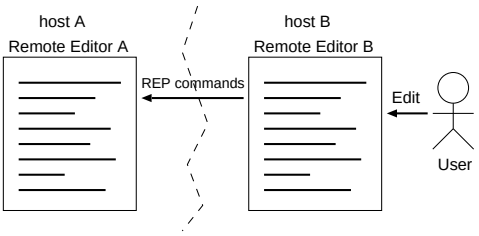

図 2 単方向のファイル編集

編集前に Remote Editor A と Remote Editor B のバッファの内容が一緒だとする。ユーザが Remote Editor B へ行った編集を、REP コマンドへ変換し、同じ順番で Remote Editor A へ反映させると、編集後に Remote Editor A と Remote Editor B のバッファが一致することは明らかである。

2.3 双方向編集

双方向編集の場合は図 3 のように互いの編集を REP コマンドへ変換し、相手が発した REP コマンドを自分のバッファへ反映させる。

しかし、双方向のテキスト編集では以下の 2 点が問題となる。

(1) 独立した編集による編集対象領域の行番号のずれ
(2) REP コマンド列の発行元が複数あるので、受け取る REP コマンド列のシーケンスがエディタ毎に異なってしまう。

以下にこの問題を説明する。

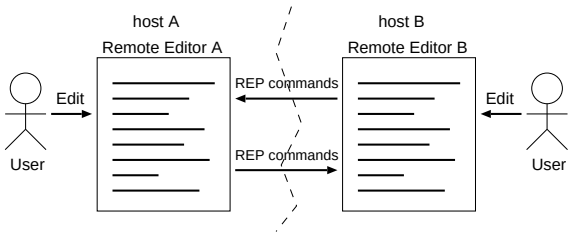

図 3 双方方向のファイル編集

2.4 行番号のずれ

双方向の編集ではそれぞれが独立して行の削除・挿入を行うので、編集対象の領域の行番号にエディタ間でずれが生じる場合がある。図 4 を見て欲しい。Remote Editor A と Remote Editor B でそれぞれ `delete 3` と `insert 2 NEW_line` という編集をしたとする。その編集を REP コマンドにして相手に送信し、同じように編集しても、 Remote Editor B の側では行の挿入により `delete 3` の編集対象とする行の行番号が Remote Editor A と異なっており、編集後のテキストの内容が異なってしまう。

これらのずれを修正するために、リモートエディタには REP マージャという二つの REP コマンド列を比較し、ずれを補う機能がある。それを用いることにより、他のリモートエディタから受け取る REP コマンドと、自分が生成した REP コマンド列による編集のずれを修正するような変換を行うことができる。

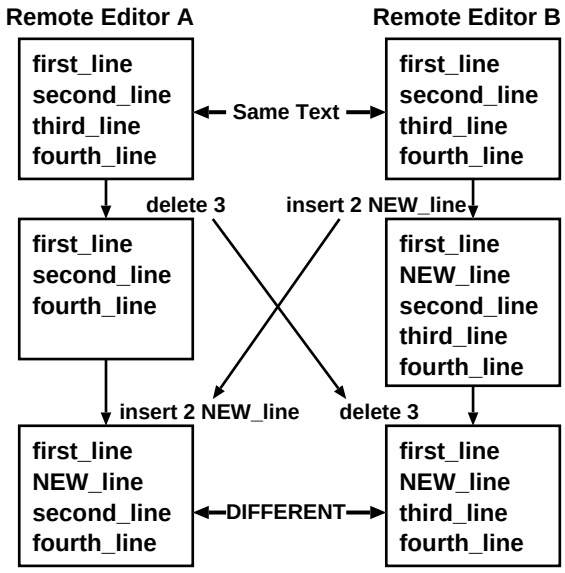

図 4 双方向編集の行番号のずれ

2.5 エディタごとの REP コマンド列のシーケンス

双方向のテキスト編集では REP の編集コマンドの発行元が複数になる。すると、各リモートエディタが受け取る他のリモートエディタの REP の編集コマンド列の順番が受け取るタイミングによって異なるので、各リモートエディタ間のテキストバッファの整合性が保持できない。例を挙げると、図 5 で、Remote Editor A が Remote Editor B, C からそれぞれ `delete 3`, `insert 2 NEW_line` という REP コマンドを受け取ったとする。Remote Editor A はこの REP コマンドによる編集を自分のバッファに対して行わなければならないが、`delete 3`, `insert 2 NEW_line` の順序を入れ換えて実行してしまうと、結果が違ってしまう。よって、テキストバッファの整合性が保てない。

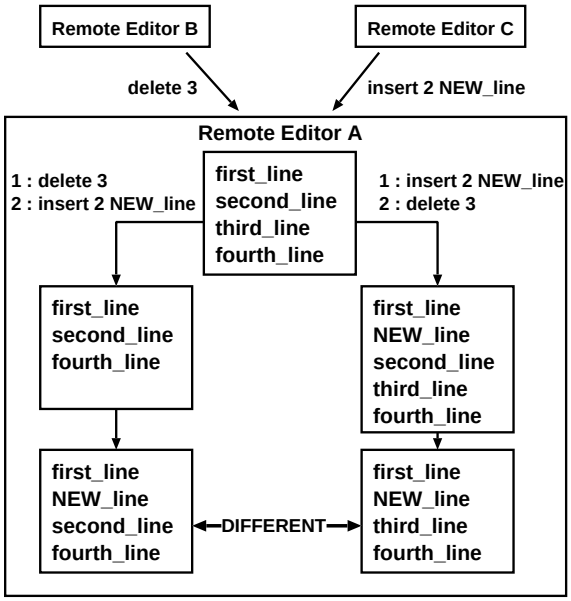

図 5 双方向編集の REP コマンド列のシーケンスのずれ

この問題を解決するために、リモートエディタが出力した REP コマンド列の順番を一元化する必要がある。リモートエディタでは、この一元化の手法として巡回トークンを用いる。巡回トークンを用いることにより、リモートエディタが受け取る REP コマンド列は一意に決まる。

3. 巡回トークン

この章では巡回トークンを用いて一元化した REP コマンドと自ら生成した REP コマンドのずれを修正し、トークンへ統合 (マージ) する REP マージャについて説明する。また、トークンを巡回させるリングネットワークについても説明する。

ここで述べるトークンとは、リモートエディタが生成した REP コマンドを他のリモートエディタへ送信できる「送信権」と捉えると分かり易い。つまり、REP コマンド列をひとつにまとめてトークンとし、それを各リモートエディタ間を巡回させる。各リモートエディタはトークンを受け取ったら、自分が生成した REP コマンドをトークンへ書き込む。そしてこのトークンを他のリモートエディタへ送信する。この方法を用いると、全リモートエディタで受け取る REP コマンド列のシーケンスを一意に定めることができる。

3.1 REP マージャ

REP コマンドは行番号によってその編集箇所を指定する。REP コマンド列はリモートエディタ毎に独立して生成されおり、生成した REP コマンドが編集する行が 他のリモートエディタでは異なる行番号にある可能性がある。

この行番号のずれや編集の競合などを REP コマンド列の比較・変換によって修正し、テキストバッファの整合性を保つのが、REP マージャである。

REP マージャは リモートエディタが生成した REP コマンド列と受け取ったトークンを入力として受け取る。そして自身のテキストバッファへの編集を行う REP コマンド列と、自身が生成した REP コマンド列を統合したトークンを出力する (図 6)。

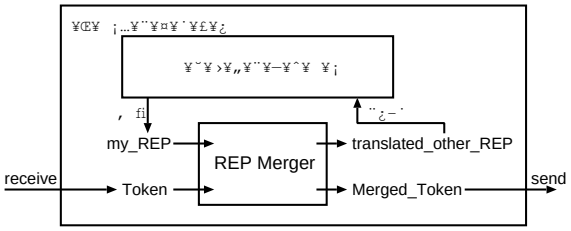

図 6 REP マージャ

以下からは REP マージャが行う変換について説明する。

3.1.1 異なる行番号における変換

図 7 におけるリモートエディタ A,B の `delete 3` の REP コマンドは行番号は同じだが、削除する行は違う。これは、エディタ B ではまず `insert 1 INS_B` を行っているため、テキストバッファの行番号とテキストの対応が変化しているためである。このような場合、単純に行番号を比較することはできない。

この問題を解決するには、比較する REP コマンドの前の REP コマンドによる行番号の変化を相手に反映させなければならない。つまり、図 7 において、リモートエディタ A の REP コマンドの `delete 3` に リモートエディタ B の REP コマンド `insert 1 INS_B`

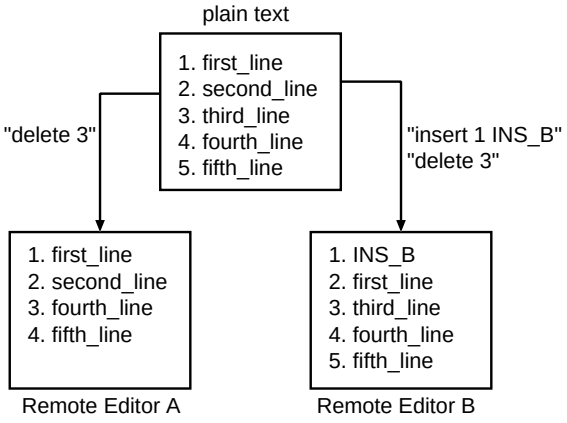

図 7 エディタ列の比較

によるテキストバッファの行番号の変化を反映させる。すると、`insert 1 INS_B` によって行番号は 1 増えているので、リモートエディタ A の `delete 3` は、リモートエディタ B の `delete 4` と等しくなる。`insert 1 INS_B` の行番号は `delete 3` の行番号より小さいので、`delete 3` の行番号を 1 増やす。つまり、

- 行番号の大きい方の REP コマンドの行番号を、
- 行番号の小さい方の REP コマンドが insert なら +1 delete なら -1 する

というような作業を繰り返す。プログラムで書くと以下のようになる。ここで、REP コマンドの列を A, B とし、それぞれの列のコマンド を a, b とする。

```
for (REP コマンド列 B) {
    for (REP コマンド列 A) {
        if (a の行番号 > b の行番号) {
            if (REP コマンド b が insert) {
                a の行番号に 1 を足す
            }
            if (REP コマンド b が delete) {
                a の行番号から 1 を引く
            }
        } else if (a の行番号 < b の行番号) {
            if (REP コマンド a が insert) {
                b の行番号に 1 を足す
            }
            if (REP コマンド a が delete) {
                b の行番号から 1 を引く
            }
        } else if (a の行番号 == b の行番号) {
              ...
        }
    }
}
```

この作業により、リモートエディタ間の独立した編集による、REP コマンドの行番号のずれを修正することができる。

ここで、行番号が重なったときのプログラムは省略している。編集対象の行が重なった場合の競合を解消する変換については、次で説明する。

3.1.2 行番号が一致した場合の変換

上で説明したマージは REP コマンドを比較したときに行番号が異なるときの処理である。これは行番号を比較したときに、比較した相手の REP コマンドによって行番号を増減させる。しかし REP コマンドの行番号を比較した場合、行番号が等しいときのマージはまた違った処理が必要になる。

REP コマンドが等しいときのマージは、優先順位によって競合を解消する。この優先順位は、リモートエディタに割り振られた番号などにより決定する。この場合の競合の解消で、REP コマンドの変更や REP コマンドの削除などの変換が行われることもある。REP コマンド A,B の組み合せは以下の 9 通りである。

- A が insert,B が insert
- A が insert,B が delete
- A が insert,B が replace
- A が delete,B が insert
- A が delete,B が delete
- A が delete,B が replace
- A が replace,B が insert
- A が replace,B が delete
- A が replace,B が replace

以上の組み合せ全てにおいて優先順位に基づいて REP コマンドの変換を行う。リモートエディタ A が発行した REP コマンド A が優先順位が高いとする。REP コマンド列 A の変換規則は表 1, REP コマンド列 B は表 2 に挙げる。

- 0 – なにもしない
- +1 – 行番号を +1
- i – コマンドを insert にする
- X – コマンドを削除する

		REP コマンド B		
		insert	replace	delete
REP コマンド A	insert	0	0	0
	replace	+1	0	i
	delete	+1	X	X

表 1 REP コマンド列 A の変換

REP コマンド列 の変換について説明する。行番号が重なったときの処理として、この変換ではなるべくテキストが残るような変換を行っている。

まず、片方が insert のときには新規に行が追加されるだけなので、既存の行への影響は行番号の変化のみである。例えば REP コマンド A が insert なら、リモートエディタ A のテキストバッファでは REP コマンド B の編集対象の行は 1 行増やした行番号に移動していることになる。よって、REP コマンド B の行番号を 1 増やす。双方が insert の場合、優先度が低い方の行を増やすことにより、お互いの編集内容が一致する。

		REP コマンド B		
		insert	replace	delete
REP コマンド A	insert	+1	+1	+1
	replace	0	X	X
	delete	0	i	X

表 2 REP コマンド列 B の変換

次に、片方が replace のときについて説明する。replace は対象となる行のテキストを別のテキストで上書きするコマンドである。このとき問題となるのが、対象行に対して相手が編集を加えていたときである。例えば、REP コマンド A が replace のときに REP コマンド B も replace なら、どちらかに合わせなければならない。これは、優先度の高い方 (ここでは A) に合わせる。また、REP コマンド B が delete のとき、リモートエディタ B のテキストバッファでは上書きするはずの行が削除されている。このときはなるべくテキストを残すように、replace を insert にし、相手の delete をキャンセルするようにする。

delete は insert 以外のときは、キャンセルされてしまう。変換規則が、テキストをなるべく残すようになっているためである。また、両方の REP コマンドが delete のときは、両方ともキャンセルする。これは、削除する行は、すでに相手先で削除されているとお互いに分かるからである。

3.1.3 マ ー ジ

これらの変換処理を経て独立した編集による REP コマンド列の編集領域のずれを修正する。この修正が終わると、変換された トークン内の他のリモートエディタの REP コマンド列を自身のテキストバッファへ反映し、変換された自身の REP コマンドはトークンへ統合する。

統合は、REP コマンド列の変換が終わっており競合箇所がないので、トークンが保持している REP コマンド列の最後尾に付け加えることができる。REP マージャの行う統合作業は以上である。

この統合作業を各リモートエディタごとに行うことで、複数のリモートエディタで独立した編集を行っても、テキストの整合性を保つことができる。

3.2 リング型ネットワーク

トークンを各リモートエディタへ配信するトポロジとして、リング型ネットワークを用いる。リモートエディタをノードとし、リング状に接続して各リモートエディタ間をトークンが巡回する (図 8 参照)。

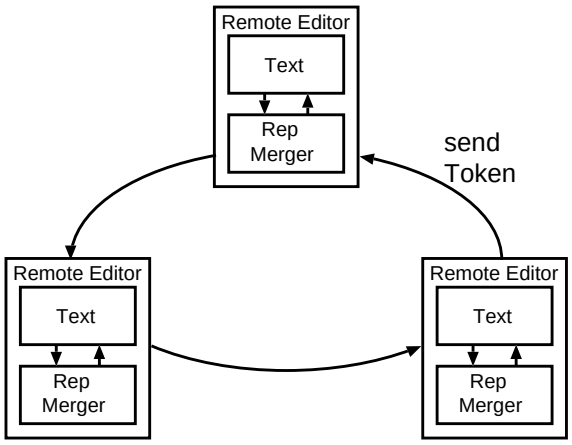

図 8 リングネットワーク

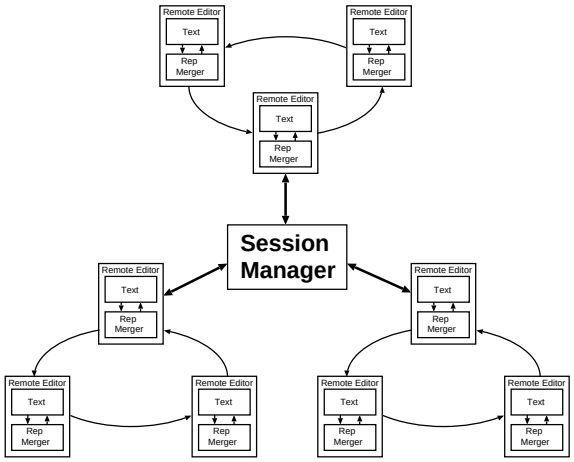

図 9 セッションマネージャとリング

リングネットワークを用いる利点として以下ような点が挙げられる。

- 通信量が少ない
- 全リモートエディタの REP コマンド受信を確認しやすい

ネットワークを流れるデータがトークンのみなので、通信量が少ない。送り先と受け先が 1 つずつというシンプルな構造なので、実装が容易である。

トークンに書き込んだ REP コマンドが一周すると、全リモートエディタがその REP コマンドを受け取ったということが分かる。これは、不要になった REP コマンドの削除を容易にする。実際、トークンを受け取る際には自分が発行した REP コマンドは破棄するようにしている。

4. セッションマネージャ

ここではセッションマネージャについて説明する。

セッションマネージャは編集 (セッション) 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、現在行われているセッションを管理する。クライアント (ここではリモートエディタ) はセッションマネージャに接続し、セッションバッファに自分の情報を書き込むことにより、セッションが行われているリングネットワークへ参加することができる。

リモートエディタはセッション単位でリング状に接続されるが、それらのリングは全てセッションマネージャと接続している (図 9)。トークンはリング内を巡回するのでセッションマネージャを介した REP コマンドによる通常の編集は行われることはない。しかし、リングへの参加などのセッションバッファに対する処理の場合は、セッションマネージャが REP コマンドを発することがある。

以下にセッションバッファとセッションマネージャの主な機能であるセッション参加について説明する。

4.1 セッションバッファ

セッションバッファはセッションに参加しているリモートエディタの IP アドレス、ポート番号、バッファ名を保持するテキストバッファである (図 10)。リモートエディタはこのセッションバッファからリングネットワークを構築するので、セッションバッファに自分の IP アドレスやポート番号などの情報を書き込むことにより、セッションに参加することができる。セッションバッファもテキストなので、編集には REP を用いる。

セッションバッファはセッションごとに用意されているので、セッションマネージャは複数のセッションバッファを持つことになる。

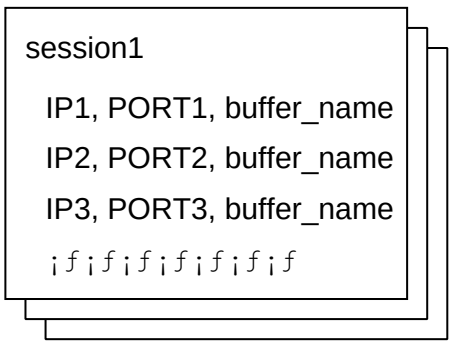

図 10 セッションバッファ

4.2 セッション参加要求とリング形成

以下に新規にリモートエディタを加える手順を示す(図 11)。

(1) 新規リモートエディタが接続要求を出す
(2) セッションマネージャからセッションバッファを取得
(3) 参加したいセッションバッファに IP アドレスなどを書き込む
(4) REP を用いてセッションバッファの変更がセッションマネージャに伝わる
(5) セッションマネージャを通じて、セッションに参加しているリモートエディタにセッションバッファの編集が伝わる
(6) リモートエディタは変更されたセッションバッファから、新規リモートエディタを加えたリングを形成する。

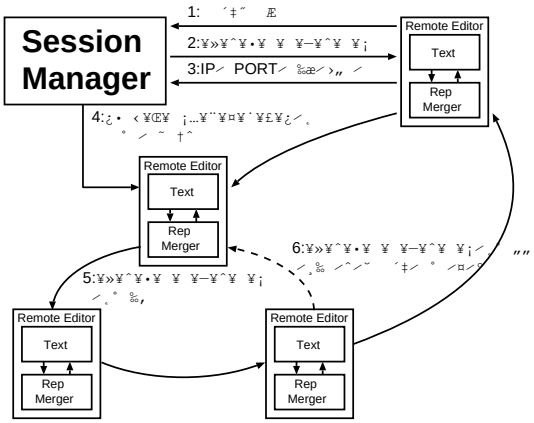

図 11 セッションへの参加

このように、リング形成もセッションバッファを REP 用いて書き換えることにより実行できる。セッションバッファを用いることにより、リングネットワークの欠点であるノードの切断にも柔軟に対応することができる。

5. 考　　察

本稿では巡回ノードを用いた REP コマンド列の一元化と REP マージャによって、リモートエディタを用いた複数人テキスト編集を可能にした。

今回 REP マージャの変換時に「なるべくテキストが残るよう」な変換を行った。これは、REP マージャでもユーザの意図する競合解消は難しく、最終的なテキストの競合修正はユーザに委ねることにしたためである。実際、自分が編集している箇所と他のユーザが編集している箇所が重なった場合にはユーザ同士でしかどの編集がベストなのか判断できない。しかし、優位な権限をあるユーザに与えて、そのユーザが行った編集を最優先するようなこともできる。教育現場などでは教師にそのような権限を与えて、生徒に指導するというようなことにも使える。これらの変換時の権限の変更などは、オプションとして選択できることが好ましいと考えられる。

巡回ノードを用いて各リモートエディタ間を巡回させる他に、サーバを設けて REP コマンド列を一元化する方法も考えられる。これは、SOBA を用いて Eclipse に P2P 機能を付加するプラグインである Sobalipse が取っている形式である[6)7)]。サーバを用いると管理が容易になるが、編集するたびにサーバへアクセスしなければならず、サーバへの負荷が集中してしまう。また、サーバが落ちてしまうと編集作業自体がストップしてしまう。巡回ノードとセッションマネージャを用いたリングネットワークは、負荷を分散させ、ネットワークの切断に強いことが分かる。

6. まとめと今後の課題

本稿では巡回トークンを用いて REP コマンド列を一元化することにより、複数人の独立した編集のシーケンスを一意に定める方法を提示した。また、REP マージャを用いた編集対象領域のずれや競合の解消について説明した。

今後はより多くのアプリケーションへ REP を実装するための API の開発を進めていきたい。

参　考　文　献

1) 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
2) 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
3) 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
4) 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
5) 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
6) Soba. http://www.soba-project.org/jp/index.html
7) sobalipse. http://sobalipse.sourceforge.net/