リモートエディタとそのXMLへの応用
Remote Editing Protocol and it's application to XML

新垣 将史[†]　　河野 真治[†]
Masasi Arakaki　　Shinji KONO

[†]琉球大学理工学研究科情報工学専攻
Speciality of Information Engineering, University of the Ryukyus.

概 要

リモートエディタは、異なるマシン上のファイルをリモートシェルやNFSなどを介さずに、直接接続して編集を行うことにより、分散環境下での効率的なテキスト編集を行うものである。これを拡張可能なマークアップ言語XMLに応用し、構造データを効率的に遠隔操作する方法を提案する。

1　はじめに

これまで、ネットワークを介した異なるマシン上に保存されているファイルを編集する手法一つとしてリモートエディタを提案し、その通信プロトコルの設計と実装を行ってきた。リモートエディタの特徴として、サーバ・エディタが複数のユーザのファイルを管理するので、他のユーザが作成したファイルの利用が容易なことや、大きなファイルの編集でもネットワークへの負荷が少ない、といったことが挙げられる。

本稿ではリモートエディタをXML文書の編集に特化し、その編集データを効率的に通信するためのプロトコルについて設計、実装を行う。

2　リモートエディタ

2.1　リモートエディタの概要

リモートエディタとは、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に接続し編集を行うエディタである。

リモートエディタでは、編集作業時のデータ送受信は、サーバ側で管理するファイル全体のバッファと、クライアントで管理するウインドウ(画面)にファイル内容を表示するためのバッファ間で行う。ファイル入出力はサーバ側でローカルに行われ、クライアントはサーバのバッファと通信することによりファイル編集を行う。

リモートエディタでは、すべてのファイルをサーバで管理することにより、他のクライアントが編集したファイルに対して追加編集を行うといったことができる。そのため、多人数での書類の編集などが容易になることが期待できる。

2.2　既存のリモートエディティングとの比較

ネットワークによって繋がる異なるマシン上のファイルを編集する既存の方法として、NFSを用いた編集とリモートログインを用いた編集が挙げられる。

ここでNFSを用いた編集とは、ローカルマシン上で起動したエディタによって、NFSマウントしたマシン上のファイルを編集することとする。また、リモートログインを用いた編集とは、rloginやtelnetなどによりネットワーク上の他のマシンへログインし、そのマシン上のエディタを用いて編集することとする。

NFSを介した編集の問題点として、ファイル入出力の際に、そのデータ全てがネットワークを介して転送されることである。ファイルのオープンのた

びにクライアントへファイルデータ全てを転送するので、ネットワークやクライアントへの負荷が大きくなる。

リモートログインを介したファイル編集の問題点としては、キー入力のたびに通信を行うため、ネットワークの速度が低下した場合、その影響が編集作業に直接影響することが挙げられる。またこれは、X ウィンドウシステムで、X サーバと X クライアント間の通信でも同様の問題が言える。

リモートエディタでは、クライアントのバッファには編集に必要なデータ (ウインドウに表示する分のテキストデータなど) を持ち、必要に応じてサーバ・エディタのバッファと通信を行うことで編集を行う。これにより NFS のようにファイル入出力時にファイルデータをすべて転送するような状況は少なくなる。また、編集データはある程度まとめて転送することで、編集作業中のネットワークトラフィックを削減し、編集作業の効率化を図る。

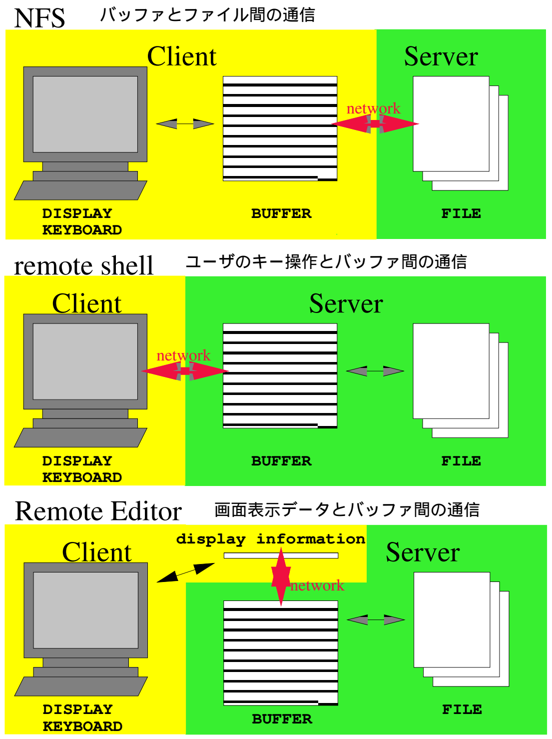

図 1　NFS、リモートログイン、リモートエディタによるファイル編集

2.3　これまでの実装と評価

これまで作成したリモートエディタのプロトタイプでは、通信プロトコルに RPC(リモート・プロシジャ・コール) を用いた。また、ベースとなるエディタとしてワシントン大学により開発された PICO を用いた。エディタ間のプロトコルは、行単位で通信を行うものと、ある回数の編集処理 (文字入力、削除、カーソルの移動など) ごとに行う、2 種類を作成した。

行単位のプロトコルは、改行や行間移動ごとにその行の文字データをサーバへ送り、サーバ側のバッファの同じ行を更新することで編集結果をサーバへ反映させる。

編集処理単位のプロトコルでは、クライアントで行われた編集操作をリストに格納し、そのリストをサーバへ送る。そしてサーバは、リストを元にクライアントで行われた作業と同じ変更をサーバのバッファへ行う。

表 1 に、NFS とリモートエディタのプロトタイプによるファイル保存にかかる時間の比較を示す。NFS と比較して、リモートエディタでのファイル保存処理は、サーバ側のバッファをサーバのディスクへ書き込む処理を、クライアントから呼び出すだけなので、非常に高速である。

表 2 には、編集処理単位のプロトコルで、リスト長 (処理数) による通信時間の変化を示す。この時間は、一定の編集作業の後、キー入力と次のキー入力の間に行われる通信時間であり、これがユーザの感じる遅延時間となる。配列長 1 の場合がリモートログインを用いた編集と同等であるが、配列長による時間変化は少なく、ユーザの感じる遅延は小さいと思われる。

表 1　ファイル保存にかかる時間 (msec)

ファイルサイズ	NFS	remote editor
100KByte	534	91
1.2MByte	6188	872

表 2　リスト長による処理時間の変化 (msec)

長さ	1	10	50	100
時間	25	31	48	72

3　XMLの概要

XML(eXtensible Markup Language、拡張可能マークアップ言語)は、マーク付けされた文字をデータとして扱うことを目的としている。XMLを利用するアプリケーションは、パーサとよばれるアプリケーションによって文書中のデータを解析し、文書の要素を解析する。

XML文書は、まずXML宣言からはじまり、次にDTD(Document Type Definition、文書型定義)と呼ばれる、文書中で使用されるタグの種類、順番などの定義を記述する。DTDは、別のファイルに置くこともできる。これらの後に、DTDに従って文書のコンテンツ(インスタンス)を記述する。

XMLでは、スタイル主体のHTMLよりも厳格なタグ付けを要求する。たとえばHTMLのように終了タグが欠けていたり、タグの入れ子が重なったりすることは許されない。

正しいXML文書は、well-formed XML文書とvalid XML文書がある。Well-formed XML document は、文法やタグの対応が正しい文書である。Valid XML document は、well-formed XML文書の条件に加えて、文書のタグ付けがその文書中に記述されたDTDに従っているかどうかのチェックの済んだXML 文書である。

次にXML文書の例を示す。

```
<?XML version="1.0"?>
<register>
  <name>
    <first>...</first>
    <last>...</last>
  </name>
  <address>
    <zip>...</zip>
    <country>...
      <state>...</state>
      <city>...</city>
    </country>
  </address>
</register>
```

図2　XML文書の例

構造の注意点として、

- 開始タグと終了タグが対応していなければならない
  <foo>... </foo>(○)
- 要素が重なることを禁止している
  <foo>...<bar>...</foo>...</bar>(×)

ことが挙げられる。またこれより、XMLインスタンスは、それぞれの開始タグから終了タグまでが互いにつながり木構造を形成している。

4　設計

ここではXML文書の編集を支援するようなエディタ間のプロトコルを提案する。

XML文書を何らかの形で処理するアプリケーションは、文書を解釈するために、パーサを用いその文書がDTDに従っているかを検証し、DTDを元に文書内のデータを解析し処理を行う。つまりDTDによる検証は、XML文書を利用するアプリケーションによって文書を解釈する前に行われる。また、このテキストエディタにおけるXML文書の編集では、編集するXML文書がどのように利用されるかは特定しない。

ここでは特にXMLインスタンスの編集について考える。

サーバとクライアントのバッファ間通信において、XML文書が巨大な場合、文書のデータ全てを一度に送信するのは、ネットワークに負荷が大きく、ファイルオープンの度に編集作業が中断されることになる。XML文書の編集で、あるタグ内の要素を編集中であるとき、他のタグ内の要素全てが見える必要はない。

そこで、リモートエディタにおける編集単位をタグ単位とし、タグからタグへの移動が起こった場合に、クライアントが移動先のタグの要素およびを要求するようにし、負荷を減らすようにする。

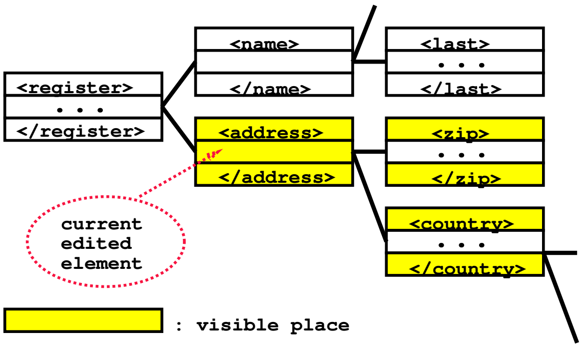

図3　ある要素の編集に必要な部分

4.1　テキスト内でのカーソル移動

クライアントからサーバへのカーソル位置の通知は、タグからタグへの移動毎に行う。

1. クライアントは、カーソル移動先のタグに含まれる要素をサーバへ要求する。
2. サーバは、クライアントから要求されたタグに含まれる要素を返す。
3. クライアントは、サーバから送られてきた要素を、画面に表示する。

4.2　新規にタグを作成する

XMLでは、"<"、">"、はマークアップ用文字であり、XMLインスタンス内ではタグを記述するのに用いられる。これら2つの文字で囲まれた文字が入力された場合を、タグの作成とする。

このとき、クライアントで行われた編集結果は、タグのネストが正確に行われた場合にサーバへ送られ、サーバのバッファへ反映されるようにする。これにより、サーバ側で管理される文書は少なくとも開始タグと終了タグの対応が正確であることが保証される。

1. クライアントは、タグが挿入されたテキスト位置をサーバへ送る。
2. サーバは、自身のバッファへクライアントから送られてきた位置をマークする。
3. クライアントは、ローカルに編集された編集結果をサーバへ送る。
4. サーバは、クライアントから送られてきたデータをマーク位置へ挿入する。

4.3　既存のタグ要素を編集する

1. クライアントは、編集するテキスト位置をサーバへ送る。
2. サーバは、自身のバッファへクライアントから送られてきた位置を含む要素をマークし、要素と、要素中に含まれるタグをクライアントへ送る。
3. クライアントは、サーバからデータ(要素)を受け取り、タグとその要素の編集が終わった後、編集したデータをサーバへ送る。
4. サーバは、マーク位置を送られてきたデータに置き換える。

5　課題

ここではXML編集を支援する機能の付加や、いくつかの問題点について考察する。

5.1　複数ユーザの利用

リモートエディタを個人で利用する場合には問題ないが、複数のユーザが一つのエディタ・サーバを利用する場合、あるユーザが作成したファイルを別のユーザが上書きしてしまう可能性がある。元のファイルはあるユーザにとっては編集中のファイルであり、それをそのまま上書きするのは好ましくない。そのため、サーバ側にはファイルのリビジョンコントロールが必要となる。

5.2　セキュリティの問題

ファイルの編集は、許可されたユーザによって行われるべきであるため、ユーザの認証が必要になる。ユーザ認証には、/etc/passwdファイルなどを利用する方法も考えられるが、リモートエディタのユーザごとにそのホストにユーザアカウントを作成しなければいけないため、効率的ではない。よって、リモートエディタ側に独自の認証システムを持つことを考えている。

5.3　編集支援機能の追加

4節では、XML文書の編集データを効率的に転送する方法を考察したが、編集の際に文書の要素など全体像が見えにくくなるという問題がある。そこで、文書内のタグのみを表示するプロトコルや、タグによる検索などの機能の付加などで、それら問題を補間することも重要な課題である。

参考文献

[1] John Bloomer著、"RPCプログラミング"、アスキー、1995

[2] “Extensible Markup Language(XML)1.0”、http://www.w3.org/TR/1998/REC-xml-19980210

[3] Michael Leventhal、David Lewis、Matthew Fuchs著、"XML 畝計技法"、アスキー、1999

[4] David Megginson著、"XML文書の構造"、アスキー、1999