# 分散フレームワーク Christie を用いた remote editor
# Remote editor using distributed framework Christie

165713F 一木貴裕　指導教員：河野真治

## 1　複数人によるファイルの同時編集

情報社会の発展に伴い、特定の場所に赴かずとも仕事を行うことができるリモートワークや、互いのいる場所を問わず画面越しに対話が行える遠隔会議といった取り組みが注目されている。

これらの取り組みをより発展させる方法として remote editor の開発を行うことにした。remote editor とは異なるマシン上の text editor を接続しリアルタイムに同期することでひとつのファイルを複数人で操作するものである。プログラミング教育や共同開発に行われる手法としてペアプログラミングが挙げられる。remote editor を実装し、共通のファイルを複数人で同時に操作を可能にすることによりペアプログラミングの効能をより高められると考えた。しかし一人ひとりが使うプログラミングに用いられるエディタは膨大な数が存在する。共同で編集するユーザが環境を合わせる必要なく、自分の慣れ親しんだ環境で編集できるように異なるエディタ間での同期が行えるような機能を実現する。

先行研究 [1] ではネットワークをリング型で構成しトークンを巡回させていたが、ノードごとの整合性の確立が難しい、ネットワーク全体の障害に対する脆弱性の弱さといった問題点が見られた。これらの反省点を踏まえ本研究ではスター型ネットワークを用いることで remote editor の障害耐性を高める。また新しく、当研究室で開発している分散フレームワーク Christie を用いることにより、エディタ間の通信の構成を行い実用性の検討を行う。

## 2　remote editor

リモートエディタは共通プロトコルが対応するエディタが保持するバッファを開いて編集することができる。ネットワーク上の一つのバッファが編集されると他のバッファにも変更が反映され、お互いのバッファを編集し合うことができる。また、Command パターン構造となるようにプログラムを行う。エディタ間通信を行う際にコマンドを保持するという性質上 Command パターンを使う利点が生まれる。

## 3　編集位置の相違とその解消

エディタ間の通信で生じる相違について説明する。エディタ同士のコマンドの送信はそれぞれが独立して行うため、編集対象の領域にエディタ間で相違が生じる場合がある。例としてエディタが一対一の接続となっている時に発生しうる相違を図 1 を使用して解説する。

編集対象は各オフセット番号に同じ値の数字が入っているものとする。EditorA ではオフセット番号 3 の 3 という要素を削除 (テキストエディタ上のため削除されたオフセットにはその後ろの要素が繰り上げられる。)、EditorB ではオフセット番号 2 に A という要素を挿入するという編集をしたとする。この編集を共通プロトコルとして互いに送信しあった際、本来編集する予定だったオフセットの中身が食い違ってしまい最終的に異なった内容となってしまう。これらの問題を解決することのできるエディタ同士の通信方法を確立しなければならない。

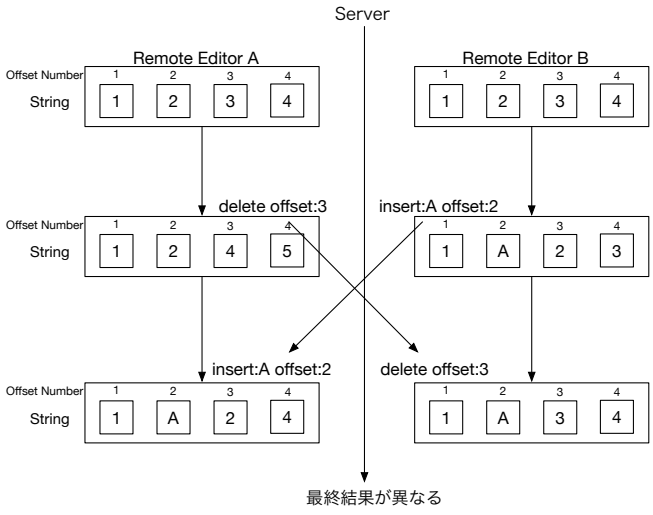


図 1: 通信によるオフセット位置のずれ

## 4 document listener による編集オフセット番号の読み取り

エディタ同士の基本通信環境の構成のため、java 言語で作成したエディタを使い異なるマシン同士の同期を実現する。自作エディタは java.swing の機能で構成されており、追記もしくは削除されたオフセット位置とその内容の取得は DocumentListener を使用している 。

```
textArea.getDocument().addDocumentListener(new
     MyDocumentListener());

class MyDocumentListener implements DocumentListener
     {

    public void insertUpdate(DocumentEvent e) {
        Document doc = (Document)e.getDocument();
        int loc = e.getOffset();
        System.out.print("location␣=␣" + loc + "\n");
        try {
            System.out.print("string␣=␣"+ doc.getText(
                 loc, 1) + "\n");
        } catch (BadLocationException e1) {
            e1.printStackTrace();
        }
    }

    @Override
    public void removeUpdate(DocumentEvent e) {
        Document doc = (Document)e.getDocument();
        int loc = e.getOffset();
        int e_length = e.getLength();
        int del_loc_end = loc + e_length - 1;
        if(e_length == 1) {
            System.out.print("delete␣" + loc + "\n");
        }else{
            System.out.print("delete␣" + loc + "␣to␣" +
                  del_loc_end + "\n");
        }
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
    }


}
```

## 5 スター型ネットワーク

異なるマシン上の editor 同士を接続するスター型ネットワーク解説する。

スター型とはネットワーク接続形態の一つであり、主要となるサーバー (ハブ) から接続された他の全てのノードが直接接続される形のネットワークトポロジーである。一般的な LAN はスター型で接続されており、またハブ同士を接続したり tree 型にノードを構成するといった自由性もある。

先行研究のリング型ネットワークと比較したスター型ネットワークの利点として

- リング型ではエディタ同士の変更内容の一元化が難しいが、スター型ではサーバーが中心となるため一元性の保持が容易である。
- ノードが障害を起こしても影響がそのノードのみに限られる。また、再接続の際はサーバーを参照することで可能となる。

が挙げられる。

## 6 分散フレームワーク Christie

ここでは当研究室が開発している分散フレームワーク Christie について説明する。

Christie はユーザーが分散プログラムを行う際、並列で動く資源などの複雑性を緩和しながらプログラムを書き上げることができる構造となっている。接続された異なるノード間において互いのキーの差し合いだけで通信を行うことができ、remote editor の通信を手軽に実現することができる。また、Christie は java 言語で開発されている。また同じく当研究室で開発している言語 Continuation based C(以下 CbC) で構成されている GearsOS に組み込まれる予定がある。そのため CbC と似た Gear というプログラミング概念が存在する。Gear は以下の四種類が存在する。

- CodeGear (CG)
- DataGear (DG)
- CodeGearManager (CGM)
- DataGearManager (DGM)

CodeGear はクラス、メソッドに相当し、DataGear は変数データに相当する。CodeGearManager はノードであり、CodeGear, DataGear, CodeGearManager を管理する。DataGearManager は DataGear を管理するものであり、put という操作により変数データ、つまり DataGear を格納できる。

DataGearManager には Local と Remote の二種がある。Local であれば、Local の CodeGearManager が管理している DataGearManager に対し、DataGear を格納する。Remote であれば、Remote 先の CodeGearManager に DataGear を格納する。

また、DataGear はアノテーションを付けデータの取り出し方を指定する必要がある。アノテーションには Take と Peek の二つがあり、Take は読み込んだ DataGear を保持せず消えるが、Peek は DataGear を保持し続ける。また、RemoteTake,RemotePeek というものもあり、リモート先を指定することにより、Remote の DataGearManager からデータを取ることができる。

CodeGear は CodeGearManager によって実行される。ただし、CodeGear 内に記述された DataGear が全て入力される必要がある。もし DataGear が揃わない場合、CodeGearManager は DataGear が揃うまで待機状態となる。

## 7 今後の課題

現時点では Christie と同じ java 言語で作成したエディタを作り、一対一のエディタ同士の通信を確認している。また、現在はオフセット番号を取得し同期を行なっているが、

既在のエディタは行単位での挿入と削除を行なっているため、将来的には行単位に統一する必要がある。

## 参考文献


[1] 安村恭一. 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, Merch 2004.

[2] 河野真治. 分散フレームワーク christie と分散木構造データベース jungle. IPSJ SIG Technical Report, May 2018.

[3] Kaito TOKKMORI and Shinji KONO. Implementing continuation based language in llvm and clang. *LOLA 2015*, July 2015.

[4] 照屋のぞみ. 分散フレームワーク christie の設計, March 2018.

[5] 宮城健太. Remote editing protocol の実装, Merch 2008.

[6] 結城浩. デザインパターン入門, 2004.