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　研究背景

近年、情報社会の発展にともないプログラミングを始めとした IT 技術に対する注目が集まっている。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の増加、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、小学校教育の一環にプログラミングを取り組むといった動きがその一例と言える。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。

開発する remote editor は異なるマシン上の text editor を接続し、異なるエディタ間でも通信が行えるよう編集コマンドを統一する共通プロトコルを用いる。接続された一つのマシン上のエディタで編集を行うと、編集位置と内容を逐次、共通の編集コマンドに変換する。変換されたコマンドを接続ネットワークに送信することで遠隔でのテキスト編集を行う。

この remote editor は先行研究が存在する [1]。先行研究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、芳しい結果が得られなかった。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 の高速化を計る。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、Christie の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。

3　共通プロトコルのコマンド

共通プロトコルの編集コマンドは insert と delete の二種類しか存在しない。この二種のコマンドを用いてエディタ間の協調動作を実現する。

- insert: バッファに挿入された内容とオフセット位置をキーとし、他のバッファへ送信し反映させる。
- delete: バッファで削除されたオフセット位置をキーとし、他のバッファへ送信し反映させる。

4　編集位置の相違とその解消

共通プロトコルとエディタ間で生じる相違の解消について説明する。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。これを防ぐためには

- 送信されたコマンドを巡回トークンを用いて一元化する。
- 自身が編集したオフセットと受信したコマンドの編集オフセットの位置を比較し、修正する。

という手順が必要となる。

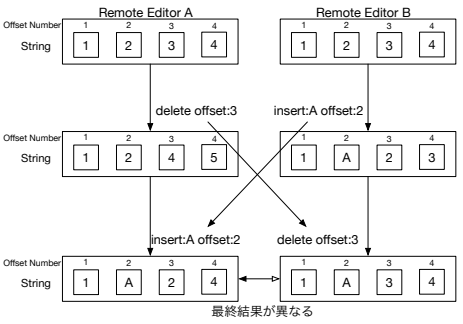

図 1: 通信によるオフセット位置のずれ

5　スター型ネットワークによる巡回トークン

スター型で構成されたネットワーク上の巡回トークンについて解説する。

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。

スター型ネットワークで接続されたリモートエディタにトークンを巡回させコマンドの送信と受信を行う。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

- リング型ではコマンドの一元化を保持するのが難しいが、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。
- リング型では速度が問題となり遅延が発生していたが、スター型にすることにより改善が期待できる。

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. IPSJ SIG Technical Report, 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.