情報処理学会論文誌 Vol.53 No.2 1–5 (Feb. 2012)

VNCを用いた授業用画面共有システムの設計・開発

谷成 雄[1,a)] 大城 信康[1] 河野 真治[1]

受付日 2011年11月4日, 採録日 2011年12月1日

概要：VNC を用いて多人数で画面共有を行う際、一つのコンピュータに同時にアクセスすると CPU 使用率やネットワークに対する負荷が高くなってしまう。そこで本研究室ではクライアントを木構造に接続させる TreeVNC を設計し、実装した。現在 TreeVNC は Multicast を用いて通信を行なっている。TreeVNC を Broadcast を用いた通信に変更するとパフォーマンスどの程度変わるのかを検証するために Broadcast を用いた通信に変更するためにはどのような変更が必要になってくるのかを考える必要がある。本研究では、TreeVNC を Broadcast で行うためにどのような変更が必要か考察し設計を行う。

キーワード：情報処理学会論文誌ジャーナル，ネットワークプロトコル，マルチキャスト，画面共有

Design and implementation of Screen Sharing System with VNC for lecture

Taninari Yu[1,a)] Oshiro Nobuyasu[1] Kono Shinji[1]

Received: November 4, 2011, Accepted: December 1, 2011

Abstract: This document is a guide to prepare a draft for submitting to IPSJ Journal, and the final camera-ready manuscript of a paper to appear in IPSJ Journal, using LaTeX and special style files. Since this document itself is produced with the style files, it will help you to refer its source file which is distributed with the style files.

Keywords: IPSJ Journal, Network Protocol, Multicast, Screen Sharing

1. 研究背景と目的

普段授業を行う際、プロジェクタなどを使って授業を進めている。しかし、後ろの席から見えにくいなどの不便を感じることがよくある。授業をうけている生徒の手元にパソコンがあるならば、そこに先生のスライドを表示して授業を進めれば後ろの席に座っても手元に画面があるので見えづらいという問題は解消される。

Ustream Producer を使用することで画面を生徒のコンピュータに配信することができる。しかし、使用してみた結果解像度が低くて、ソースコードを読む際などに見えづらいという問題が発生した。

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，情報処理大学
Presently with Johoshori Uniersity
[a)] you@cr.ie.u-ryukyu.ac.jp

他にも VNC を使えば、スライドを生徒の手元の画面に表示することができる。VNC については次の章で説明する。VNC は共有したい画面の解像度のままのデータを配信することができる。しかし、多人数の生徒が先生のパソコンに同時に接続してしまうとサーバ側が送信するデータの量が増えるので処理性能が落ちて授業の進行に画面がついていかなくなってしまう。この問題は一つのコンピュータに多人数が同時に繋がるときに起こる問題である。

そこでクライアントを Tree 構造に接続させていくことによって処理を分散させることにした。クライアントを Tree 構造に接続させることによって、見えない部分でスイッチに対する負荷が増えている分サーバに対する負荷が軽減している。

今回 TreeVNC は Multicast で実装しているが、Broadcast で実装されたものとどちらが性能がいいのかどれくらい性能差が出るのかという疑問がでてきた。

今回、Broadcast を設計する際にいろいろな課題が見えてきたので本論文では、Multicast を用いた TreeVNC の実装と Broadcast 版の設計について以下に詳しく説明していく。

2. VNC について

VNC(Virtual Network Computing) は、RFB プロトコルを用いて画面を送信して操作権を与えるリモートデスクトップソフトである。VNC はサーバ側とクライアント (ビューア) 側に分かれていて、サーバを起動し、クライアントがサーバに接続を行い遠隔操作を可能にする。

2.1 RFB Protocol

RFB (remote frame buffer) プロトコルは、GUI 操作をリモートアクセスで行うためのプロトコルである。画面の描画の更新は画面の差分が発生した部分を矩形毎で送り描画される。また、画面の描画データに使われるエンコードが多数用意されており、また独自のエンコードを実装することもできるシンプルなプロトコルである。

3. TreeVNC の方針

まず、多人数が参加している授業で VNC を使う場合に起こる問題は、最初で述べたように、一つのコンピュータに多人数が繋がり、理性能が大幅に落ちてしまうところが問題である。目的のコンピュータに繋がっているコンピュータに繋げれば目的の画面を共有することができる。

この問題を解決する為に、クライアント同士を接続させ、画面描画のデータを受け取ったクライアントが次のクライアントにデータを流すという方法を考えた。

コンピュータに繋げれば目的の画面を共有することができる。

画面共有を行いたいクライアントが一種の VNC サーバ自体にもなる。

また、クライアント同士の接続はツリー構造で行うことで管理がしやすくなると考えた。クライアント同士の接続の管理はツリーの一番上にいる PC(Top) で行い、この Top だけが VNC サーバへ接続を行うようにする。

今回作成した TreeVNC は、上記の実装でツリー状にクライアントを接続していくように実装を行った。画面の共有だけを行うように実装した。

3.1 Tree の構成

Top はクライアントが接続してくるたびに java.util.LinkedList にクライアントの情報を追加していく。

クライアントからアクセスが来るたびにスレッドを作成しているので、複数のクライアントが同時に接続してきても対応することができる。

しかし、多数のスレッドが同じ java.util.LinkedList などの共有資源に対して同時に接続を行うと共有資源の情報が正しく更新されない可能性が出てくる。

このような共有資源を更新する際は java の synchronized メソッドを使用して複数のスレッドが共有資源に同時にアクセスすることができないようにした。

Top は treeBranch(木の分木数) を定数で持っていてクライアントが接続してくるごとに counter をインクリメントしていき LinkedList の (counter - 1)/treeBranche 番目に入っている親の情報を接続してきたクライアントに教えることで木を構成することができる

3.2 Tree の再構成

木を構成することはできたが途中のクライアントが落ちてしまった場合に木を再構成しなければならない。木を再構成する手順は以下の用に行う。

(1) 子供が親が落ちたことを Top に対して報告する。
(2) Top は報告を受けると番号の一番大きいノード (最後のノード) に対して落ちた親の代わりになるように報告する。
(3) Top 命令を受けたクライアントは Top の指定された場所に接続をしなおす。
(4) Top はクライアントのリストを更新して、親が落ちた子供たちに新しい親の情報を教える。
(5) 親が落ちた子供たちは新しい親に対して接続を行う。

このようにして木を再構成することができる。以下のソースは、親が落ちた子供が接続してきた時に、子供全員の接続を待つ部分のコードの一部である。

```
final int TIMEOUT = 3000;
if (passNumber == 0) {
  passNumber++;
  wait(TIMEOUT)
  passNumber = 0;
  } else {
  if (++passNumber == TREEBRANCH) {
    notifyAll();
    passNumber = 0;
    } else {
    wait(TIMEOUT);
    }
}
```

上記のコードでは落ちた子供たちの報告を待ち合わせることができる。TREEBRANCH は木を構成する際の分木数であるフィールド変数として final で定義されている。passNumber は報告する子供たちを数えるためのカウンタである。プロキシは子供分の報告全て揃うと子供たちに新しい親の情報を流し始める。TIMEOUT が設定されているのは、子供が報告の際に落ちて報告が届かない場合があ

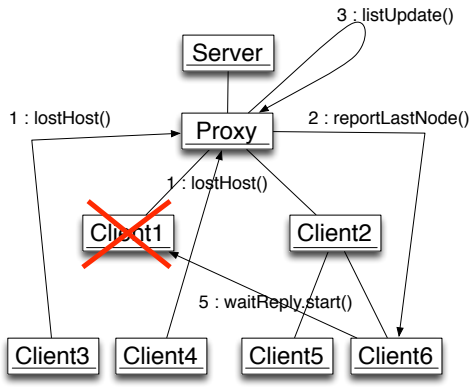

図 1 再接続の様子

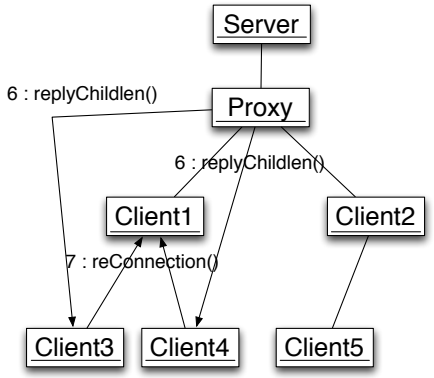

図 2 再接続の様子

るのでそれに対応したものである。

図 1 は接続の様子を記したコラボレーションダイアグラムである。以下に関数の説明をする。

(1) lostHost() は親が落ちたことを報告する関数である。

(2) reportLastNode() は番号の一番大きい (最後のノード) に対して親の代わりをするように命令する関数である。

(3) listUpdate() はプロキシが持つクライアントのリスト情報をアップデートする (落ちたノードを削除し最後のノードのアドレスをそこに追加する)。

(4) lostHost() は上記で説明したとおりである。なぜ間に 2 と 3 が呼ばれているのかというと最初に lostHost() が呼ばれて、listUpdate() が終わるまで他のノードの報告はブロックされるからである。Client4 が先に報告に行く場合もある。

(5) waitReply.start() はクライアントは waitReply というクラスを main スレッドとは別にスレッドを作成して走らせている。もしプロキシからの命令が来るとクライアントはプロキシから指定された場所に接続を行う。

(6) replyChildlen() は Listing1 で説明したとおり全クライアントが報告に来るまで待った後、親が落ちた子供たちに対して新しい親の情報を報告する関数である。

(7) reConnection() はプロキシから来た情報をもとに VNC 接続を行う関数である。

以上の関数を用いることでクライアントが落ちても木を再構成することができる。

4. リファクタリング

はじめに TreeVNC を作成する際に、Top 用のプログラムとクライアント用のプログラムを別々に作成していた。

Top とクライアントのプログラムはだいたい同じことをしているのにソースが 2 つあるので共通部分を片方を変更するすると、もう片方も同じ変更をしなければならない。片方の変更を忘れるとプログラムが正常に動作しなくなることもある。

2 つのコードを変更するのは手間がかかるので、同じ部分は一つのプログラムにまとめることにした。

リファクタリングを行う際に abstract factory を使用して 2 つのプログラムを一つにまとめることができた。

5. 圧縮の問題

VNC で扱う RFB プロトコルには、使えるエンコーディングのタイプの 1 つとして ZRLE(Zlib Run-Length Encoding) がある。ZRLE は Zlib で圧縮されたデータとそのデータのバイト数がヘッダーとして付けられ送られてくる。

Zlib はフリーのデータ圧縮及び解凍を行うライブラリである。可逆圧縮アルゴリズムの圧縮と解凍が行える java.util.zip.deflater と java.util.zip.inflater を実装している。

5.1 java.util.zip.deflater の実装の問題

Zlib 圧縮は辞書を持っていて、その辞書に登録されているデータを元に解凍が行われる。しかし、java.util.zip.deflater は現在持っている辞書を書き出すこと (flush) ができないことが分かった。辞書を書きだすことができない為、Zlib 圧縮されたデータを途中から受け取ってもデータが正しく解凍を行うことができない。

5.2 ZRLEE(ZRLE Economy)

そこで、Top が ZRLE で受け取ったデータを unzip し、データを zip し直して最後に finish() をいれることで初めからデータを読んでいなくても解凍を行えるようにした (毎回新しい辞書を使うようにした)。

このエンコードは ZRLEE エンコードと定義した。一度 ZRLEE エンコードに変換してしまえば、そのデータをそのまま流すだけで良い。よって変換は Top が行う一回だけですむ。

ただし、deflater,inflater では前回までの通信で得た辞書をクリアしないといけないため、Top とクライアント側では毎回新しく作る必要がある (クライアント側は inflater だけ)。また、ZRLEE はクライアント側が対応していなければならないという問題がある。

5.3 ZRLE と ZRLEE のデータ圧縮率の比較

RAW,ZRLE,ZRLEE のデータ量の比較を行った。図 6 は 1920 * 1080 の画面の全描画にかかるデータ量を測った結果を示した図である。ZRLEE の方がデータ量が少なくですんでいる。これは、ZRLE(Zlib) が初めに送られた辞書を用いての解凍が余り有効的に働いていない場合があるからだと思われる。つまり VNC の場合は ZRLEE の様に毎回辞書のデータを付加させて送ってもデータ量に差がでない可能性があることが分かった。

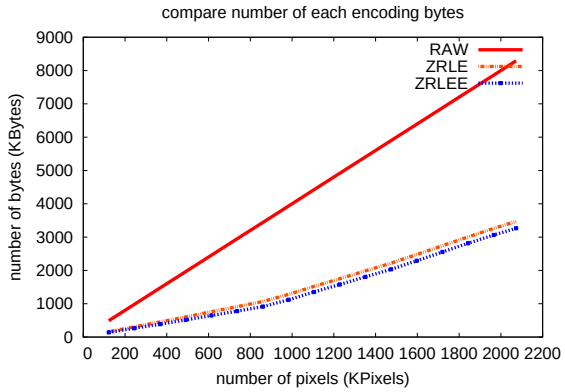

図 3 RAW,ZRLE,ZRLEE による 1 画面 (1920*1080) 描画にかかるデータ量。x 軸はピクセル数、y 軸はバイト数を表している。

6. 授業などでの使用

実際の授業で実装した TreeVNC を使用してみた。実際に使ってみての感想などを聞いてみると良かったなどの意見がでてきた。しかし、実際に使ってみて新しい問題などもみつかった。

TreeVNC では各クライアントが自分自身のタイミングで画像データを取得できるように MulticastQueue を作成して、データを Queue にもたせている。

クライアントが接続を持ったまま suspend(停止) してしまうと Queue にデータがたまり続けてしまい Memory Overflow を起こしてしまう。そこで、TimeOut スレッド作成して、一定時間取得されないデータは Queue から削除して、クライアントが読み込みを開始した際に、消された次に入っているデータを送るというように設計されている。

一つ目の問題は画像の更新が少ない画面の共有を行う際に、たまにデータが送られてこないとう問題がある。この問題はおそらく上記で説明した TimeOut スレッドが更新データを落としてしまっているのではないかと考えられる。

二つ目の問題は発表者が複数いる際は発表者が変わるごとに Top を起動しなおさなければならないということである。この問題については次の章でもう少し詳しく説明する。

7. UserInterface の設計と実装

普通 VNC で接続を行う際にクライアントを起動する際に相手のアドレスを指定しなければならない。

そこで、TreeVNC はクライアントが起動する際に Broadcast パケットを送信して同じセグメント内に、Top が起動しているかどうかを調べ、起動していたら起動している Top の一覧を表示するように設計・実装を行った。

これによりクライアントは起動する際に何も打ち込まなくても起動することができる。

前章で説明したように、ゼミなど発表者が多数いる際に、TreeVNC を使用すると発表者ごとに Top を立て直す必要がある。このような際にボタンひとつで、画面共有を行う対象を変更できると便利である。しかし、VNC では違う画面サイズのデータが流れてくると落ちてしまう仕様になっている。

そのため、画面を切り替える際はクライアントに画面が切り替わることを教えるプロトコルを作成する必要がある。上のような設計で画面の切り替えは今後実装する予定である。

8. Broadcast 版の設計

この章では Broadcast 版を設計する際に見えてきた問題点などを解説する。

8.1 Broadcast パケットの性質

Broadcast パケットの性質とし大きすぎるデータの送信などができないという性質がある。実際に Broadcast パケットでどの程度の大きさのデータが送れるのかを測ってみたところ有線の Broadcast だと約 64000byte までのデータ量だと送信することができた。無線の Broadcast の場合も約 64000byte のデータを送信することができたが、無線だと不安定でデータが沢山落ちたりすることもあった。

もう一つの問題は Broadcast パケットの性質としてデータが落ちてもわからないという性質があり更新データが落ちたのをクライアント側が気づくことができないという問題である。この問題に対しての考えは後述する。

8.2 パケットの分割

VNC では画像を更新する際に矩形単位で更新データを送信する。

画面全体の更新などはどうしても更新データのサイズが大きくなってしまう。Broadcast では約 64000byte までのデータしか送ることができないので、データを送信する際は 64000byte 以下になるようにデータを分割して送らなければならない。

第 5 章で説明したとおり TreeVNC では、VNC Server から ZRLE 圧縮されたデータをうけとり、Top が一旦解凍をして、圧縮し直して ZRLEE 圧縮されたデータを送信している。

この一旦解凍されたデータを分割 64000byte 以下にして

クライアントに送信してやれば良い。ZRLE を解凍すると、64*64 ピクセルのタイル群のデータになる。

よってデータの分割はこのタイル群数個分で分割することでうまくいくと考えた。最後に分割したデータを圧縮しなおして、header データを付加してクライアントに流してやれば良い。

8.3 drop したパケットの検出

Broadcast の性質で説明したとおり、Broadcast ではデータが落ちていることをクライアントが知ることができない。そこでプロトコルを拡張してデータごとにシリアル番号を振ってやり連続でない値が来た時はデータが正しく届いていないと判断することができる。このようにすることで drop したデータの検出ができると考えた。

8.4 Acknowledge の設計

データが drop した際に、どのようにして落ちた部分のデータを Top に再送してもらうかが問題となってくる。クライアントは drop したパケットのシリアル番号がわかっている。

ちなみに、Broadcast 版の実装でもクライアントを Tree 構造で接続させておく必要がある。その理由として、初期接続のパケットは Broadcast では送れないので TCP を用いなければならないやクライアントが drop を検出したとき Top に知らせて drop した部分のパケットを再送信してもらわなければならない。再送信をする際は TCP で送信する。

ここで Tree 構造で組んでいない場合、全クライアントが TOP に対して接続に行ってしまうと接続が一箇所に集中してしまい通常の VNC と変わらなくなってしまう。

以上の理由から Broadcast の際にも Tree 構造で組む必要がある。そうすると、Tree 構造でどのようにして drop したパケットを Top に教えるかを考えなければならない。

クライアントは上のクライアントに対してどの番号のパケットが drop したのかを指定して最終的に Top に届くように上へ上へと送信していく。ここで、上へ送信していく際に途中で目的のデータが上から送られてきた際には、そのパケットはそこで破棄するようにする。

9. まとめと今後の展開

本研究では、作成した TreeVNC の UserInterface の設計を行い実装を行った。さらに、TreeVNC は Multicast でパケットを送信しているが、Broadcast で送信することができるようにするための設計を行った。設計を行うことでいろいろな問題が見えてきた。

今後の展開として、発表者が変わるたびに Top を立て直すまたはひとつのコンピュータで発表を行うなど不便な部分があるので発表者をボタンひとつで切り替えられるように実装を行いたい。

それから、今回設計を行った Broadcast 版の実装を行い、Multicast 版の TreeVNC と Broadcast 版の性能比較を行いたい。

参考文献

[1] TightVNC: VNC-Compatible Free Remote Control / Remote Desktop Software: http://vnc-reflector.sourceforge.net/
[2] TightVNC: VNC-Compatible Free Remote Control / Remote Desktop Software: http://www.tightvnc.com/
[3] Tristan Richardson, Quentin Stafford-fraser, Kenneth R. Wood, Kenneth R. Wood, Andy Hopper: Virtual Network Computing (1998): Virtual Network Computing (1998)
[4] P. Deutsch, J-L. Gailly : ZLIB Compressed Data Format Specification version 3.3
[5] 谷成雄, 大城信康, 河野真治 : VNC を用いた授業用画面共有システムの設計と実装 日本ソフトウェ ア科学会第 28 会大会, Sep 2011