解　説

VNCを用いた授業用画面共有システムの設計と実装

河野 真治　谷成 雄　大城 信康

各クライアントを Tree 型に接続し, 親が配信したデータをリレーさせることで分散 VNC アプリケーションを実装した. 通常の VNC では配信者へ負荷が集中する設計となっている. 例えば, 大学の講義等で VNC を用いて画面共有を行った時, クライアントの増加に比例して配信者への負荷が増えてしまう. この問題を解決する為に, Tree 構造にクライアントを接続させ, Top のクライアントから子供へデータを送ることでスケーラビリティを持たせた. その結果, クライアントの数を増やしてもサーバ側への負荷を抑えることができた. また, VNC Refrector との性能比較も行う.

1　はじめに

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。　VNC（画面共有）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。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。この問題は一つのパソコンに多人数が繋がいるときに起こる問題である。　本論文では、多人数で画面共有ができるようにクライアントをツリー構造に接続させ、上から順番にデータを流していくという方法で新しい VNC の設計・実装を行う。

Design and implementation of Screen Sharing System with VNC for lecture
Shinji Kono, Yu Taninari, Nobuyasu Oshiro, 琉球大学 工学部情報工学科 並列信頼研究室, Dept. of The Department of Information Engineering, University of Ryukyus Concurrency Reliance Laboratory.
コンピュータソフトウェア, Vol.27, No.0 (2010), pp.1–5.
[解説論文 (レター)] 2010 年 1 月 7 日受付.

2　実装方法

2.1　tree structure

今回は、ホストに対しクライアントがツリー状に繋がっていくように実装した。ツリーの構成は以下の手順で行う。

1. クライアントが接続する際、ホストに接続をしているプロキシ（今後このプロキシのことをトップと記述する）に接続する。
2. トップはクライアントにどこに接続すれば良いかを知らせる。（このときに親の番号と自分の番号それからリーダーであるかどうかを一緒に知らせる）
3. クライアントはトップから指定されたノードに接続を行う。

2.2　tree の再構成

今回の実装はクライアントがツリー状に繋がっているので、親ノードが落ちると子ノードも一緒に落ちてしまう。そこで、tree の再構成が必要になる。

1. 親ノードが落ちた際に、子ノードの中で一番若い番号の子ノードがトップに対して自分の親ノードが落ちたことを報告する。（親ノードの番号を

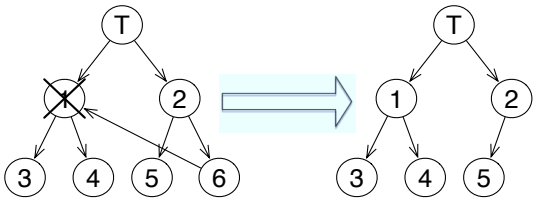

図 1　１番の木が落ちたときの再接続の処理 (T は TOP)

知らせる）

2. トップは木の番号が一番大きいノードに対して１で報告を受けた親ノードの代わりになるように命令を出す。

3. 親ノードがいなくなった子ノードたちはトップに対して、２で新しく繋がった親ノードの IP アドレスを教えてもらいそのアドレスに対して接続をおこなう。上記の構成の場合、一つのノードが落ちた場合に再接続を行うノードは２分木の場合３ノードである。

3　java.util.zip.deflater のバグ

VNC で扱う Rfb Protocol には、使えるエンコーディングのタイプとして ZRLE(Zlib Run-Length Encoding) がある。ZRLE は Zlib 圧縮されたデータを内包する。deflater はプリセット辞書をもち、Zlib 圧縮されたデータはその辞書を用いて解凍が行われる。辞書は更新されることもあるので Zlib 圧縮されたデータを解凍する為には辞書のデータも受け取る必要がある。しかし、Java にはこの Zlib の辞書を相手へ書きだす (flush) する機能が無い。元々の Zlib の規約にはこの辞書を flush する機能があったが Java には実装されていなかった。これは Java.util.zip.deflater のバグである。

4　ZRLEE

そこで、Top Proxy 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。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 Proxy が行う一回だけですむ。ただし、deflater 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、Client 側では毎回 deflater は新しいものを使うことになる。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　ZRLE と ZRLEE のデータ圧縮率の比較

ZRLE と ZRLEE を用いて通信を行う場合、データ量にどれくらいの差がでたのかを図 2 に示す。図 2 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 のデータの中には deflater が持つ辞書のデータを更新しようとするため 1 つの BufferedUpdate のたびに辞書を送信するはずの ZRLEE が優っているのは、VNC がこれは、VNC では Zlib で圧縮されたデータを解凍する際に、持っていた解凍の為の辞書がそこまで役に立たないことを示している。ZRLE

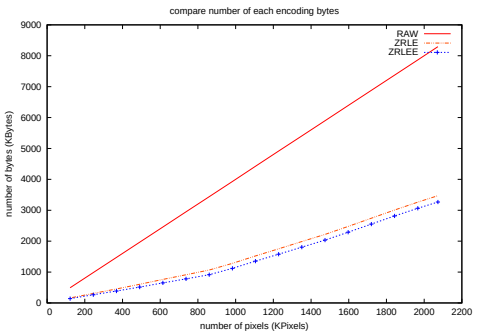

図 2

6　参考文献の参照

7　参考文献リスト

8　謝辞

謝辞は，参考文献の前に，次のように書く．

```
{\bf 謝辞}\ 本論文の初期の版について議論していただいた A 氏に感謝する．
```

参 考 文 献

[1] Lamport, L. : *A Document Preparation System LaTeX User's Guide & Reference Manual*, Addison-Wesley, Reading, Massachusetts, 1986.