第57回 プログラミング･シンポジウム 2016.1.8-10

有線 LAN 上のPC画面配信システムTreeVNCの改良

伊波 立樹[1,a)] 河野 真治[1,b)]

概要：授業やゼミ等で、それぞれが PC 端末を持っている場合では、PC の機能を活かしたコミュニケーションが可能である。教員が操作する画面をそのまま学生に配信したり, ゼミなどで、発表する学生の画面を切り替えたりすることを可能にしたい。画面配信システム TreeVNC は参加したクライアントをバイナリツリー状に接続し、配信コストを分散させる仕組みを取っている。そのため, 多人数が参加しても処理性能が下がらない。また、ツリーのルートが参照している VNC サーバーを変更することで、ケーブルの差し替えなしに画面の切替が可能となる。本研究では TreeVNC の改良として、複数のネットワークへの対応、WAN への対応、 マルチディスプレイへの対応を行うとともに、TreeVNC 有用性を示すために画像データの遅延時間計測を行った。

キーワード：VNC, 分散アプリケーション

1. 画面共有を利用したコミュニケーション

授業やゼミ等で、それぞれが PC 端末を持っている場合では、PC の機能を活かしたコミュニケーションが可能である。

通常の授業では先生の用意した資料、 PC 画面を見ながら授業が進むことが多い。ゼミでは発表者を切り替えながら発表を行う。

通常これらの画面を表示するためにプロジェクタが使用されている。しかし、プロジェクタでは通常の授業の際、参加者はプロジェクタに集中するため、手元の PC をほぼ使用することができない。更に手元の PC を使う際はプロジェクタと PC を行き来するため、目に負担がかかってしまう。またゼミの際には発表者を切り替えるたびにプロジェクタにケーブルを差し替える必要がある。ケーブルの差し替えの際に発表者の PC によってアダプターの種類や解像度の設定によって正常に PC 画面を表示できない場合もある。

画面配信システム TreeVNC[1] は発表者の画面を参加者の PC に表示する。そのため、参加者は不自由なく手元の PC を使用しながら授業を受ける事が可能になる。更に発表者の切り替えの際もケーブルの差し替えずに共有する画面の切替を可能としている。

Tree VNC は VNC[2] を使用した画面配信を行っている。しかし通常の VNC では配信側に全ての参加者が接続するため、多人数の際の処理性能が落ちてしまう。Tree VNC では有線でネットワークに接続した参加者をバイナリツリー状に接続し、 配信コストをクライアントに分散させる仕

[1] 琉球大学工学部情報工学科
[a)] innparusu@cr.ie.u-ryukyu.ac.jp
[b)] kono@ie.u-ryukyu.ac.jp

組みになっている。そのため、授業で先生の画面を表示する際、多人数の生徒が参加しても処理性能が下がらない。また、ツリーのルートが参照している VNC サーバーを変更することで、共有する画面の切替が可能となる。

しかし TreeVNC を授業やゼミで使用している中、 様々な問題が発生した。TreeVNC は起動した PC が複数のネットワークに接続していても、単一のネットワークしか使用することが出来ない。更に、NAT を越えたネットワーク接続に対応しておらず、遠隔地などで授業やゼミに参加することが出来ない。また、ゼミの際に、マルチディスプレイを使用して画面配信を行う際、すべての画面が配信され、不必要な画面まで表示されてしまう。

そこで、本研究では上記の問題点を解決し、TreeVNC の有用性を評価することで授業やゼミを円滑に行えることを目標とする。

2. 画面配信システム TreeVNC

2.1 VNC について

VNC(Virtual Network Computing) は、 RFB プロトコルを用いて遠隔操作を行うリモートデスクトップソフトウェアである。VNC はサーバー側とクライアント (ビューア) 側に分かれている。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とする。

2.2 RFB プロトコル

RFB(remote frame buffer) プロトコル [3] とは、自身の画面を送信し、ネットワーク越しに他者の画面に表示するプロトコルである。ユーザが居る側を RFB クライアント側と呼び、Framebuffer への更新が行われる側は RFB サーバと呼ぶ。Framebuffer とは、メモリ上に置かれた画像データのことである。RFB プロトコルでは、最初にプロトコルバージョンの確認や認証が行われる。その後、クライアントに向けて Framebuffer の大きさやデスクトップに付けられた名前などが含まれている初期メッセージが送信される。RFB サーバ側は Framebuffer の更新が行われるたびに、RFB クライアントに対して Framebuffer の変更部分だけを送信する。更に RFB クライアントの FramebufferUpdateRequest が来るとそれに答え返信する。RFB プロトコルは、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プロトコルである。

2.3 多人数で VNC を使用する時の問題点

VNC を使用すればクライアント側にサーバー側の画面を表示することが可能である。しかし、多人数のクライアントが 1 つのサーバーに接続してしまうと処理性能が落ちてしまうという問題点がある。

また、 ゼミ等の発表で画面配信者が頻繁に切り替わる場合、配信者が替わる度にアプリケーションを終了し、接続をし直さないといけないという問題がある。

2.4 TreeVNC の構造

TreeVNC は Java を用いて作成された TightVNC(Tight Virtual Network Computing)[4] を元に作成されている。

TreeVNC は 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方式を取っている。また、サーバへ接続しに来たクライアントをバイナリツリー状に接続する (図 1)。バイナリツリー状に接続することで、N 台のクライアントが接続しに来た場合、画面配信の画像データをコピーする回数は従来の VNC ではサーバ側で N 回する必要があるが、TreeVNC では各ノードが 2 回ずつコピーするだけで済む。

TreeVNC で通信される画像のデータ量は大きいため、大きなネットワークスループットが必要である。そのため、有線接続が必須である。

バイナリツリーのルートのノードを Root Node と呼び、 Root Node に接続されるノードを Node と呼ぶ Root Node は子 Node にデータを流す機能に加え、各 Node の管理、 VNC サーバーから流れてきた画像データの管理を行う。Node は 親 Node から送られたデータを 自分の子 Node に流す機能、 逆に子 Node から送られてきたデータを

親 Node に流す機能がある。

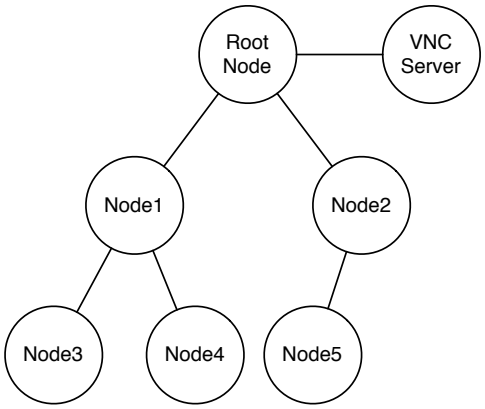

図 1 構成される木構造

TreeVNC は ZRLEE[5] というエンコードでデータのやり取りを行う。ZRLEE は RFB プロトコルで使えるエンコーディングタイプの ZRLE を元に生成される。

ZRLE は Zlib[6] 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て送られてくる。Zlib は java.util.zip.deflater と java.util.zip.inflater で圧縮と解凍が行える。

しかし、java.util.zip.deflater は解凍に必要な辞書を書き出す (flush) ことが出来ない。辞書を書き出すことが出来ないため、 Zlib 圧縮されたデータを途中から受け取ってもデータを正しく解凍することが出来ない。

そこで ZRLEE は 一度 Root Node で受け取った ZRLE のデータを 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呼んでいなくても解凍を行えるようにした。

一度 ZRLEE に変換してしまえば子 Node はそのデータをそのまま流すだけで良い。ただし、deflater と 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、Root Node と Node 側では毎回新しく作る必要がある。

2.5 TreeVNC の通信経路

TreeVNC の通信経路として以下が挙げられる

- ある Node から Root Node に直接通信を行う send direct message (Node to Root)
- Root Node からある Node に直接通信を行う send direct message (Root to Node)
- Root Node から木の末端の Node までのすべての Node に通信を行う messeage down tree (Root to Node)
- ある Node から木構造を上に辿って Root Node まで通信を行う message up tree (Node to Root)
- Root Node から配信者の VNC サーバーへの通信を行う send message (Root to VNC-Server)
- VNC サーバーから Root Node への通信を行う send message (VNCServer to Root)

2.6 ノード間で行われるメッセージ通信

RFB プロトコルで提供されているメッセージに加え、 TreeVNC 独自のメッセージを使用している。TreeVNC で使用されるメッセージの一覧を表 1 に示す。

表 1 通信経路とメッセージ一覧

通信経路	message	説明
send direct message (Node to Root)	FIND_ROOT	TreeVNC 接続時に Root Node を探す。
	WHERE_TO_CONNECT	接続先を Root Node に聞く。
	LOST_CHILD	子 Node の切断を Root Node に知らせる。
send direct message (Root to Node)	FIND_ROOT_REPLY	FIND_ROOT への返信。
	CONNECT_TO_AS_LEADER	左子 Node として接続する。接続先の Node が含まれている。
	CONNECT	右子 Node として接続する。接続先の Node が含まれている。
message down tree (Root to Node)	FRAMEBUFFER_UPDATE	画像データ。EncodingType を持っている。
	CHECK_DELAY	通信の遅延を測定する。
message up tree (Node to Root)	CHECK_DELAY_REPLY	CHECK_DELAY への返信。
	SERVER_CHANGE_REQUEST	画面切り替え要求。
send message (Root to VNCServer)	FRAMEBUFFER_UPDATE_REPLY	画像データの要求。
	SET_PIXEL_FORMAT	pixel 値の設定。
	SET_ENCODINGS	pixel データの encodeType の設定。
	KEY_EVENT	キーボードからのイベント。
	POINTER_EVENT	ポインタからのイベント。
	CLIENT_CUT_TEXT	テキストのカットバッファを持った際の message。
send message (VNCServer to Root)	FRAMEBUFFER_UPDATE	画像データ。EncodingType を持っている。
	SET_COLOR_MAP_ENTRIES	指定されている pixel 値にマップする RGB 値。
	BELL	ビープ音を鳴らす。
	SERVER_CUT_TEXT	サーバがテキストのカットバッファを持った際の message。

図 2 は TreeVNC で Node が Root Node に接続し、画像データを受信するまでのメッセージ通信の様子である。図 2 の手順として

- 接続を行う Node (以下 Client Node) は Multicast 通信で Root Node に対して FIND_ROOT を送信する (1:findRoot())
- Root Node が FIND_ROOT を受信し、

FIND_ROOT_REPLY を送信する (2:findRootReplay())
- Client Node 側で、どの Root Node に接続するかを選択するパネルが表示される
- Client Node はパネルで接続する Root Node を選択し、Root に対して接続先を要求する WHERE_TO_CONNECT を送信する (3:whereToConnect())
- 受信した Root Node は Client Node の接続先を CONNECT_TO で送信する (4:connectTo)
- Client Node は Root の指定した接続先に接続しに行く
- Root Node, Client Node 間の接続が確立後、Root Node から Clinet Node に対して定期的に画像データ FRAME_BUFFER_UPDATE を送信する (5:framebufferUpdate())

を行っている。

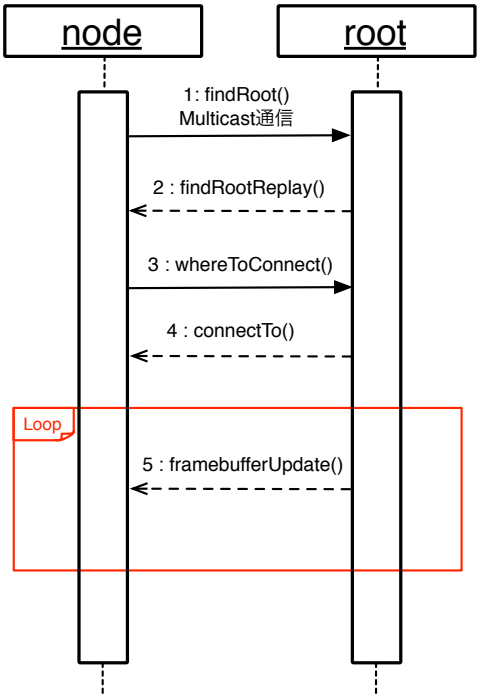

図 2 node 間で行われるメッセージ通信

2.7 切断時の木の再構成

TreeVNC はバイナリツリーでの接続という特性上 Node が切断されたことを検知できずにいると、Node 同士で構成された木構造が崩れてしまい、新しい Node が接続に来た場合に適切な場所に Node を接続することができなくなってしまう。木構造を崩さないよう、Node 同士の接続を再構成を行う必要がある。

TreeVNC の木構造のネットワークトポロジーは Root Node が持っている nodeList というリストで管理している。つまり、Node の接続が切れた場合、木の再構成を行うため Root Node に知らせなければならない。

TreeVNC は Lost_CHILD というメッセージ通信で Node の切断を検知・木の再構成を行っている。

TreeVNC は VNC サーバーから送られる画像データ (FRAME_BUFFER_Update) を MulticastQueue というキューに蓄積しており、Node はこのキューから画像データを取得し、画面を描画している。Lost_Child の検出方法はこの MulticastQueue を使用している。ある一定時間 MulticastQueue から画像データが取得されない場合 Memory Over Flow を回避するために Timeout スレッドが用意されている。Timeout を検知した際、Node との接続が切れたと判断する。

Lost_Child の検知・木の再構成を以下に示す。
- 子 Node の切断を検知した Node が Root Node へ LOST_CHILD メッセージを送信する (図 3 中, 1:lostChild())
- LOST_CHILD メッセージを受け取った Root Node は nodeList の更新を行う (図 3 中, 2:updateNodeList())
- 切断した Node を nodeList から消し、nodeList の最後尾の Node に切断した node number を割り当てる
- Root Node は最後尾の Node に、切断した子 Node が接続していた親 Node に接続する様に CONNECT_TO メッセージを送信する (図 3 中, 3:connectTo(1))
- 最後尾の Node が子 Node を失った親 Node へ接続しに行く (図 3 中, 4:connectToParent(1))

LOST_CHILD によって、 切断された全ての

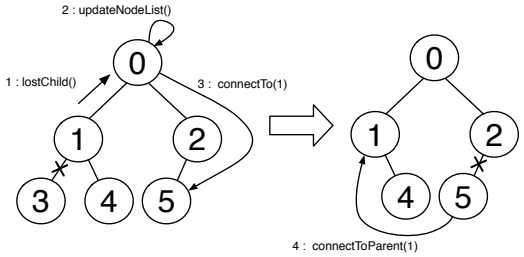

図 3 LOST_CHILD を検知・再接続

Node を検知することができるため、 nodeList の更新が正しく行われる。よって、新しく接続に来た Node を適切な場所に接続することが可能となる。

2.8 共有画面切り替え

ゼミでは発表者が順々に入れ替わる。発表者が入れ替わる度に共有する画面の切り替えが必要となる。ゼミを円滑に進めるために、画面の切り替えをスムーズに行いたい。

画面の共有にプロジェクタを使用する場合、 発表者が変わる度にケーブルの抜き差しを行う必要がある。その際に、ディスプレイ解像度を設定し直す必要が出たり、 接続不良が起こる等の煩わしい問題が生じることがある。

従来の VNC を使用する場合、 画面の切り替えの度に一旦 VNC を終了し、発表者の VNC サーバーへと再接続を行う必要がある。

TreeVNC は配信者の切り替えの度に生じる問題を解決している。TreeVNC を立ち上げることで、ケーブルを使用する必要なしに、各参加者の手元の PC に発表者の画面を共有することができる。画面の切り替えはユーザが VNC サーバーへの再接続を行うことなく、ビューアの Share Screen ボタンを押すことによって、配信者の切り替えを行うことができる。

TreeVNC の Root Node は配信者の VNC サーバーと通信を行っている。VNC サーバーから画面データを受信し、そのデータを 子 Node へと送信している。配信者切り替え時に Share Screen ボタンが押されると、Root Node は Share Screen ボタン を押したクライアントの VNC サーバーと通信を始める。そのため TreeVNC は配信者切り替えの度に VNC を終了し、再接続する必要がない。

3. TreeVNC の新機能

3.1 QUALITY モードと SPEED モード

高解像度のまま拡大・縮小の処理を行うと、PC のスペックによっては描画処理に時間がかかってしまうことがある。配信者の画面をリアルタイムに取得するため、描画処理に時間のかからないモードを追加する。

画像描画処理には高画質優先の QUALITY モードと描画速度優先の SPEED モードがある。今まで TreeVNC は QUALITY モードで使用していた。

今回どちらのモードを使用するかをビューアから変更出来るようにした。これにより、描画処理の遅延を解決することが可能となった。

3.2 マルチディスプレイ対応

画面配信側の PC がマルチディスプレイの場合、VNC サーバーからは複数の画面全体の画像データが送信されてしまう。

授業やゼミ等で TreeVNC を使用する場合、複数画面の表示は必要ない。そこで、画面を共有する際、ディスプレイを選択させ、画面共有を行う機能を追加した。

ディスプレイの情報は個々のクライアントでしか取得ができない。そのため、配信側は画面の切替を行う際に、ディスプレイを選択し、そのディスプレイの左上と右下の座標を取得する。その座標を Root Node への画面切り替えを要求する SERVER_CHANGE_REQUEST メッセージに付加させる。Root Node は 配信側の VNC サーバー に画像データを要求する FRAMEBUFFER_UPDATE_REPLY メッセージに送信された座標を付加する。VNC サーバーは要求された座標内の画像データを FRAMEBUFFER_UPDATE メッセージで Root Node に送信する。これにより、一画面のみの表示が可能となる。

図 4 は Display1 のみを画面共有する例を示している。

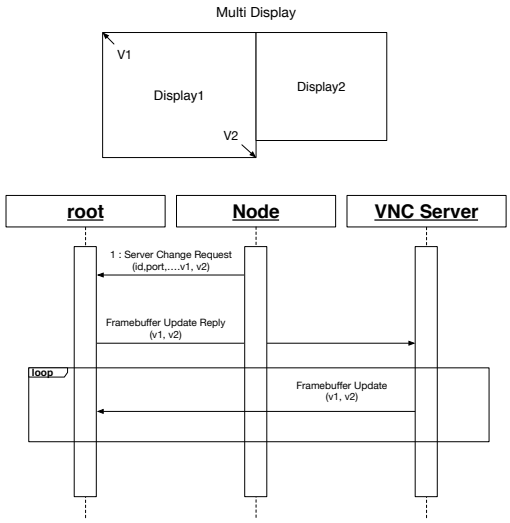

図 4　マルチディスプレイへの対応

3.3　複数のネットワークの対応

従来の TreeVNC は、クライアントの接続する木構造が単一であった。そのため、Root Node が複数のネットワークに接続していても、 単一のネットワークでしか使用することができなかった。

この問題を解決するために、 図 5 の様に、ネットワーク別に 木構造を形成するように設計した。

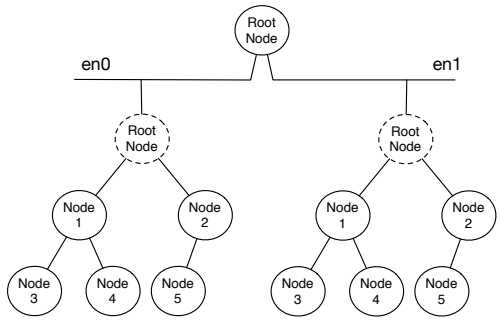

図 5　Multi Network Tree

TreeVNC は Root Node が TreeManager というオブジェクトを持っている。TreeManager は TreeVNC の接続部分を管理している。TreeManager では木構造を管理する nodeList が生成される。この nodeList を元に、新しい Node の接続や、切断検出時の接続の切り替え等を行う。

Root Node の保持しているネットワーク毎に TreeManager を生成する用に変更を行った。新しい Node が接続してきた際、 interfaces から Node のネットワークと一致する TreeManager を取得し、 Node 接続の処理を任せる。そのため、TreeVNC が複数のネットワーク別に木構造を構成することが可能となる。

3.4　WAN への対応

遠隔地からでもゼミや授業に参加できるよう、NAT を越えたネットワークから TreeVNC への接続を可能にした。

図 6 に NAT を越えたネットワークからの接続を示す。別ネットワークから TreeVNC に参加する際、 直接配信側のネットワークの Root Node に接続を行う。この接続を Direct Connection と呼ぶ。

Direct Connection した Node はそのネットワークの Root Node になる。そのネットワークの他の Node はそのネットワークの Root Node に接続し、木構造を生成する。

配信側の Root Node は Direct Connection で接続された Root Node に対して Framebuffer Update で 画像データを送信する。Framebuffer Update が送信された Root Node は そのネットワークの Node に対して Framebuffer Update を送信する。

これにより、NAT を越えたネットワークの画面共有が可能となる。

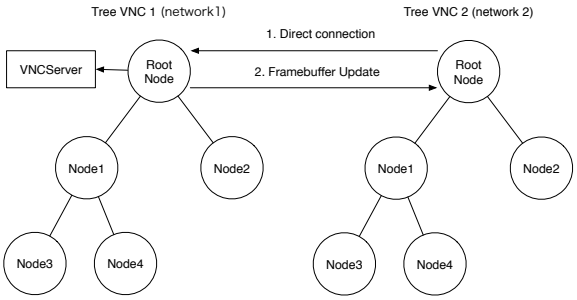

図 6　遠隔地 Node からの接続

4. TreeVNC の評価

4.1 木の深さによる画像データ伝達の遅延

VNC サーバー から受信する画像データ、TreeVNC で扱われるメッセージ通信は構成された木を伝って伝達される。接続する人数が増える毎に木の段数は増えていく。 そこで Root Node から木の末端の Node までの画像データ伝達の遅延を検証する実験を行った。

4.2 実験環境

授業を受講している学生が TreeVNC を使用した状態で実験を行った。TreeVNC には最大で 17 名が接続していた。

4.3 メッセージを使用した実測

TreeVNC を伝搬するメッセージに、CHECK_DELAY,CHECK_DELAY_REPLY を追加した。CHECK_DELAY は Root Node から 末端の Node まで伝達するメッセージと画像データ、CHECK_DELAY_REPLY は各 Node から Root Node まで伝達するメッセージである。

CHECK_DELAY メッセージは送信時刻を付けて送信する。Root Node から CHECK_DELAY 送信し、末端の Node まで各 Node を伝いながら伝達して行く。

CHECK_DELAY_REPLY は CHECK_DELAY から受け取った送信時刻をそのままに、画像データのサイズを付けて送信する。CHECK_DELAY を受け取った各 Node は CHECK_DELAY_REPLY を接続している親 Node に送信する。

CHECK_DELAY_REPLY を受け取った Root Node はメッセージと画像データの伝達にどれだけの時間がかかったかを計算する。データ計算方法を以下のソースコード 1 に記述する。 この変数 time は CHECK_DELAY_REPLY に付いている CHEKC_DELAY の送信時刻である。

4.4 深さ毎の遅延結果

バイナリツリーで木を構成した場合、 Node 数が 17 台だと深さが 4 となる。各木構造の階層毎に、画像データの伝搬にかかった時間を測定した。

```
1   Long delay = System.currentTimeMillis()
        − time;
```

Code 1 遅延時間の計算方法

図 7 は遅延の分布を示した散布図である。X 軸はメッセージ伝達にかかった秒数 (ms)、Y 軸は画像データのサイズ (Byte) である。

画像データの伝達はほぼ 1 秒以内に収まっているが、容量が小さい場合でも時間がかかる場合がある。それはその送信の前に大容量の画像を送信した後の回線の Delay が残っているためだと考えられる。

また、深さ 3 で極端に遅い場合がある。 遅い原因として、1 つの Node がボトルネックになっていることが判明している。このような極端に遅い Node をそのまま木に配置した場合、その Node の子 Node 以下に影響を及ぼす場合がある。そのため、遅い Node を検出して、木の最後尾に移動させる機能が必要である。

今回 4 段分のデータでは 30 名程度の遅延のみしか判断することができないため、更に大人数での実験を繰り返し行う必要がある。

5. 配布方法、リポジトリ

TreeVNC は jar で配布を行っている (`http://www.cr.ie.u-ryukyu.ac.jp/software/TreeVNC.html`)。

また、Mercurial でバージョン管理を行っている (`http://www.cr.ie.u-ryukyu.ac.jp/hg/Applications/TreeVNC/`)。

6. まとめ

本研究では画面配信システム TreeVNC を複数のネットワーク、 WAN、 マルチディスプレイに対応させた。

複数のネットワークに対応したことで、 PC が接続している全てのネットワークで TreeVNC を使用できるようになった。

WAN に対応することで NAT を超えているネットワークのユーザーが TreeVNC に参加すること

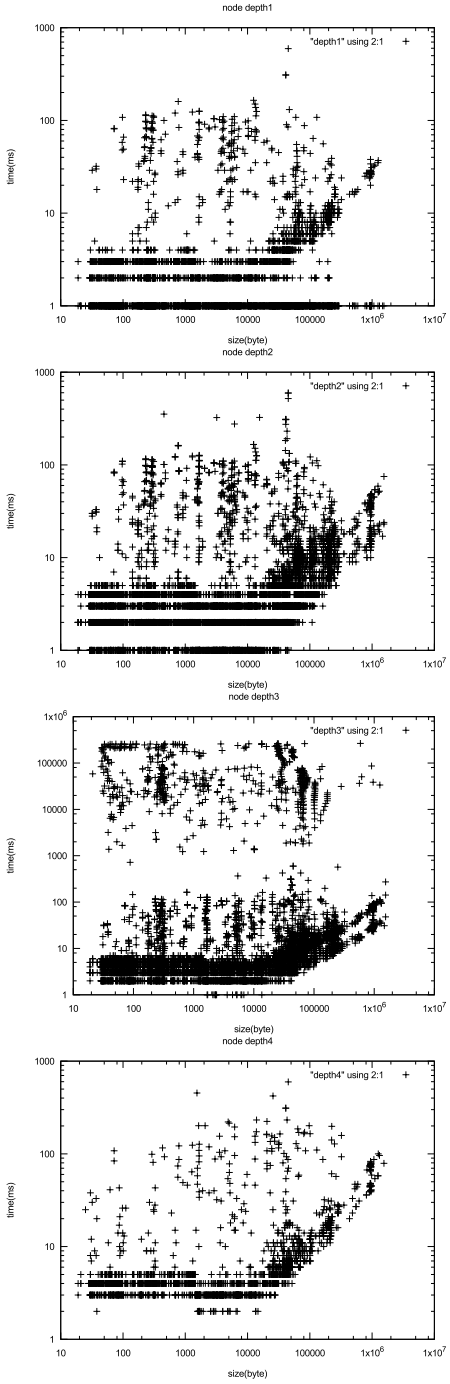

図 7 深さ毎のデータサイズと遅延の関係 (上から深さ 1, 2, 3, 4)

が可能となった。

マルチディスプレイに対応したことで、 配信者が配信したいディスプレイを選択し、画面配信することが可能となった。

今後の課題として機能の安定化、 WAN での画面切り替え、 ユーザビリティの向上、 新機能の評価が上げられる。

機能の安定化として今回の画像データの遅延実験で判明したボトルネックになる Node の対処を行う。ネックになっている Node の検出として、CHEKC_DELAY メッセージの時間を使用し、その Node がネックかどうかを判断する予定である。

今回追加した Direct Connection では NAT を越えたネットワークの画面の配信を行うのみであり、 TreeVNC の利点の 1 つである画面切り替えを行うことが出来ない。そのため、NAT を越えたネットワークでの画面切り替えの実装を行う。

Direct Connection などの一部の機能はコマンドラインオプションで指定する必要があるため、一般ユーザーでは操作するのが困難である。そこで、今までコマンドラインオプションで指定していた機能をビューアで操作するように変更を行う。

今回新機能としてマルチディスプレイ、 WAN への対応を行ったが、まだ評価を行っていない。そのため、適切な評価方法を思考し、評価を行う必要がある。

参考文献

[1] Yu TANINARI and Nobuyasu OSHIRO and Shinji KONO: VNC を用いた授業用画面共有システムの実装と設計，日本ソフトウェア科学会第 28 回大会論文集 (2011).

[2] RICHARDSON, T., STAFFORD-FRASER, Q., WOOD, K. R., AND HOPPER,: A. Virtual Network Computing (1998).

[3] RICHARDSON, T., AND LEVINE, J.: The remote framebuffer protocol. RFC 6143 (2011).

[4] TightVNC Software: `http://www.tightvnc.com`.

[5] Yu TANINARI and Nobuyasu OSHIRO and Shinji KONO: VNC を用いた授業用画面共有システムの設計・開発，情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS) (2012).

[6] LOUP GAILLY, J., AND ADLER, M.: zlib: A massively spiffy yet delicately unobtrusive compression library., `http://zlib.net`.

[7] Surendar Chandra, Jacob T. Biehl, John

Boreczky, Scott Carter, Lawrence A. Rowe: Understanding Screen Contents for Building a High Performance, Real Time Screen Sharing System, *ACM Multimedia* (2012).