画像配信システム TreeVNC のマルチキャストの導入

安田 亮[†1,a)]　大城 由也　河野 真治[†1,b)]

概要：TreeVNC とは当研究室で開発している画面配信システムである。しかし、画面共有は送信するデータ量が多いため、無線 LAN 接続の場合、画面の配信に遅延が生じてしまう。そこで、multicast でのデータ通信の実装やデータの分割・圧縮方法の評価を行い、TreeVNC の multicast の有用性を評価する。

1. 画面配信ソフトウェア TreeVNC の活用

現代の講義や発表、プレゼンなどでは PC 画面で用意した資料を見ながら進行することが多い。ゼミでは発表者の PC 画面を切り替えを行いながら発表を行う場合もある。通常このような場面では資料やスライドを表示するためにプロジェクタが利用される。その際、発表者の PC 画面を切り替えるたびにケーブルを差し替える必要がある。発表者の PC によっては接続するアダプターの種類や解像度の設定により、正常に PC 画面を表示できない場合がある。また、参加者もプロジェクタに集中を割く必要があり、手元の PC と相互に参照する場合、負担になる場合がある。

当研究室で開発している画面配信システム TreeVNC[1] は、発表者の画面を参加者の PC に表示するソフトウェアである。そのため、参加者は不自由なく手元の PC を操作しながら講義を受けることが可能になる。更に発表者の切り替えの際もケーブルを差し替えずに、共有する画面の切り替えが可能になっている。

VNC(Virtual Network Computing) は、クライアント (ビューワー) 側とサーバ側からなるリモートデスクトップソフトウェアである。遠隔操作にはサーバを起動し、クライアント側がサーバに接続をすることで可能としている。また、動作には RFB プロトコルを用いている。TreeVNC は VNC[2] を利用した画面配信を行なっている。しかし通常の VNC では配信側の PC に全ての参加者への配信を行う負荷がかかってしまう。(図 1)

†1 現在，琉球大学工学部情報工学科
Presently with Information Engineering, University of the Ryukyus.
a) riono210@cr.ie.u-ryukyu.ac.jp
b) kono@ie.u-ryukyu.ac.jp

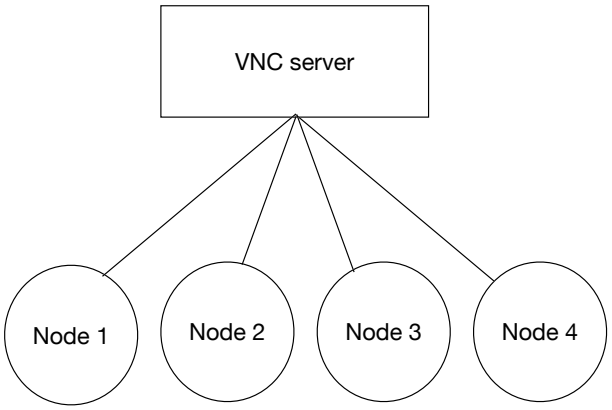

図 1　従来の VNC の接続方法

RFB(Remote Frame Buffer) プロトコル [3] とは、自身の PC 画面をネットワーク上に送信し他人の画面に表示を行うプロトコルである。画面が表示されるユーザ側を RFB クライアントと呼び、画面を送信のために Framebuffer の更新が行われる側を 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、最初にプロトコルのバージョン確認や認証が行われる。その後、クライアントへ向けて 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を送信する。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 の変更部分のみを送信する。更に、RFB クライアントの FramebufferUpdateRequest が来るとそれに答え返信する。変更部分のみを送信する理由は、更新がある度に全画面を送信すると、送信するデータ面と更新にかかる時間面において効率が悪くなるからである。

TreeVNC はサーバに接続してきたクライアントをバイ

ナリツリー状に接続する。接続してきたクライアントをノードとし、その下に新たなノード二つを接続していく。これにより、人数分のコピーと送信の手間を分散することができる。(図 2)。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きた場合、従来の VNC ではサーバ側が N 回のコピーを行なって配信をする必要があるが、TreeVNC では各ノードが 2 回ずつコピーをするだけで配信が可能となる。送信されるデータは従来の方法では N ノードに対して N-1 の通信が必要であるが、木構造を用いても通信の数は変わらない。

バイナリツリーのルートのノードを Root Node と呼び、そこに接続されるノードを Node と呼ぶ。Root Node は子 Node にデータを渡す機能、各 Node の管理、VNC サーバから送られてきたデータの管理を行なっている。各 Node は、親 Node から送られてきたデータを自身の子 Node に渡す機能、子 Node から送られてきたデータを親 Node に渡す機能がある。

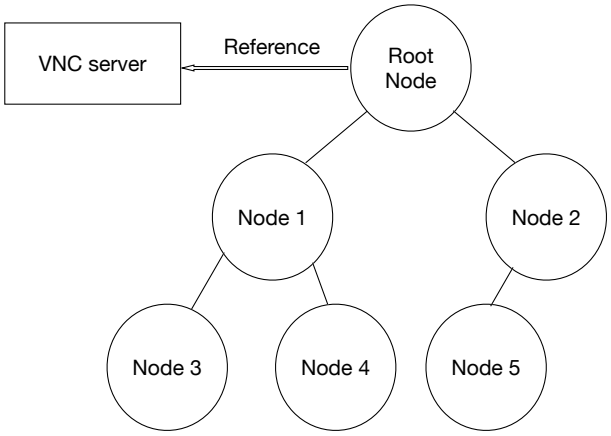

図 2　TreeVNC の接続方法

2. TreeVNC の通信プロトコル

TreeVNC の通信経路として以下の 6 つが挙げられる。

- Root Node から任意の Node に直接通信を行う send direct message (Root to Node)
- 任意の Node から Root Node に直接通信を行う send direct message (Node to Root)
- Root Node から木の末端までの全ての Node に通信を行う message down tree (Root to Node)
- 任意の Node から上に辿って Root Node まで通信を行う message up tree (Node to Root)
- Root Node から配信者への VNC サーバへの通信を行う send message (Root to VNCServer)
- 配信者の VNC サーバから Root Node への通信を行う send message (VNCServer to Root)

2.1 メッセージ通信

TreeVNC の各 Node と VNCServer 間で通信されるメッセージを表 1 に示す。

2.2 MulticastQueue

配信側の画面が更新されると VNCServer から画像データが FRAME_BUFFER_UPDATE メッセージとして送られる。その際、画像データの更新を複数の Node に同時に伝えるために Multicast Queue というキューに画像データを格納する。

2.3 木構造の再構成

TreeVNC はバイナリツリーでの接続のため、Node が切断されたことを検知できないと構成した木構造が崩れてしまい、新しい Node を適切な場所に接続できなくなってしまう。そこで木構造を崩さないよう、Node 同士の接続の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 で管理している。Node の接続が切れた場合、Root Node に切断を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 の切断を検知および木構造の再構成を行なっている。LOST_CHILD の検出方法には MulticastQueue を使用しており、ある一定時間 MulticastQueue から画像データが取得されない場合、MemoryOverFlow を回避するために Timeout スレッドが用意されている。そして、Timeout を検知した際に Node との接続が切れたと判断する。

2.4 ZRLEE

TreeVNC では、ZRLEE というエンコード方法でデータの圧縮を行う。ZRLEE は RFB プロトコルで使用できる ZRLE というエンコードタイプを元に生成される。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送信される。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 はデコードに必要な辞書を書き出す (flush) ことが出来ない (図**??**)。従って、圧縮されたデータを途中から受け取ると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 update rectangle と呼ばれる画面ごとのデータに辞書を付与して zip し直すことで初めからデータを読み込んでいなくても解凍できるようにした (図**??**)。辞書をクリアすることにより adaptive compression を実現していることになり圧縮率はむしろ向上する。

2.5 ShareScreen

従来の VNC では、配信者が切り替わるたびに VNC の

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_TO	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。

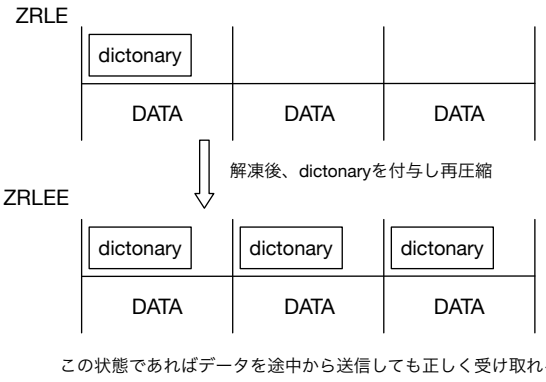

図 3　Multi Network Tree

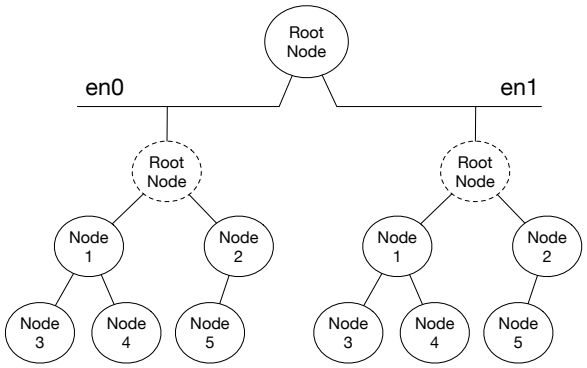

図 4　Multi Network Tree

再起動、サーバ、クライアント間の再接続を行う必要がある。TreeVNC は配信者の切り替えのた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。画面の切り替えについてはユーザが VNC サーバへの再接続を行うことなく、ビューワー側の Share Screen ボタンを押すことで配信者の切り替えが可能になっている。

TreeVNC の Root Node は配信者の VNC サーバと通信を行なっている。VNC サーバ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 を実行すると、Root Node に対し SERVER_CHANGE_REQUEST というメッセージが送信される。このメッセージには Share Screen ボタンを押した Node の番号やディスプレイ情報が付加されている。メッセージを受け取った Root Node は配信を希望している Node の VNC サーバと通信を始める。

2.6　ネットワーク複数時の接続

TreeVNC は Root Node が複数のネットワークに接続している場合、図 5 のようにネットワーク別に木構造を形成する。

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おり、木構造を管理する nodeList を生成する。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。Tree Manager は Root Node の保持しているネットワーク毎に生成される。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 Tree Manager を取得し、Node 接続の処理を任せる。

3.　マルチキャストの導入

3.1　有線接続との接続形式の違い

画面配信のデータ量は膨大なため、現在の TreeVNC で VNCServer に無線 LAN 接続を行なった場合、画面配信の遅延が大きくなってしまう。この場合でも画面切り替えの機能は有効である。つまり、画面を提供する PC のみを無線経由で接続し、配信を希望する側は有線を使用することができる。

ここで、Wifi の Multicast の機能を用いて配信側にも Wifi を使用することが可能であると考えられる。Tree Root は無線 LAN に対して、変更する UpdateRectangle を Multicast で一度だけ送信する。

Wifi の Multicast packet のサイズは 64kbyte が最大となっている。HD や 4K の大きさの画面更新は 8Mb * 8byte で圧縮前で 64MB 程度になる。これを圧縮しつつ、64kbye 毎のパケットに変換して送る必要がある。

Wifi の Multicast packet は確実に送られること保証されていない。通し番号を付けて欠落を検出することはできるが、再送処理は複雑であることが予想される。

ここではまず Blocking について考察と実験を行う。

4.　RFB の UpdateRectangle の構成

RFB の UpdateRectangle は以下の構成になっている。一つの updateRectangle には複数の Rectangle は入っていて、さらに一つ一つの Rectangle の encoding type がある。ここでは ZRLE で encode された rectangle が一つサーバから送られてくる。rectangle には zlib で圧縮されたデータが指定された長さだけ付いてくる。このデータは、64x64 の tile にさらに分割されている。tile 内はパレットとかがある場合があるが、Run length encode された RGB データである。

このパケットを 64kbyte に収まる三つの Rectangle に再構成する。この時に、tile 内部は変更する必要はないが、Rectangle の構成は変わる。ZRLE を展開しつつ、パケッ

トを構成する必要がある。

zlib はちょうど良い所で圧縮を flush する必要がある。このためには、zlib の API を用いて、適当なタイミングで flush を呼ぶ。この時に 1 tile ずづ flush してしまうと圧縮率を下げる可能性がある。

64kbyte の packet の中には複数の tile が存在するが、連続して Rectangle を構成する必要がある。行の途中から始まり、途中で終わる可能性があるので、三つの Rectangle が必要になる。

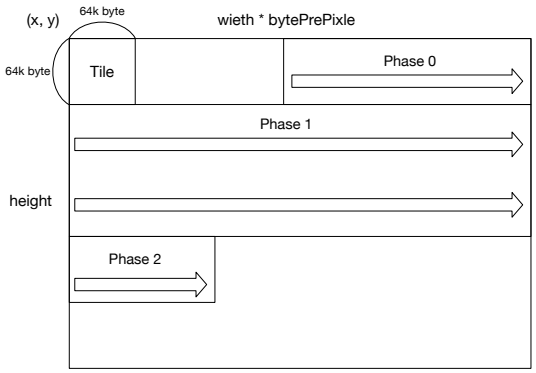

図 5 Multi Network Tree

表 **2** updateRectangle の構成

1 byte	messageID
1 byte	padding
2 byte	n of rectangles
2 byte	U16 - x-position
2 byte	U16 - y-position
2 byte	U16 - width
2 byte	U16 - height
4 byte	S32 - encoding-type
4 byte	U32 datalengths
1 byte	subencoding of tile
n byte	Run Length Encoded Tile

5. まとめ

Tree VNC に Wifi 上の Multicast packet を用いる手法について考察した。画面圧縮に Hareware supported な MPEG4 などを用いることができればより効率的な転送が可能であるが、ここでは Java 上で実装できる安易な方法をあえて選択した。Wifi の速度と Multicast の信頼性が高ければこれでも実用になると可能性がある。

Blocking は実装中であり、再圧縮の時間は前画面の時でも実用的な時間ですむことが予想されている。Wifi 上の Multicast packet の drop 率は、接続環境に依存すると思われるのでさらなる実験が必要だと思われる。

有線使用時よりも、画面共有の質が落ちるのはある程度はやむを得ないが、再送が必要である場合には、必要なプロトコルを実装する。

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] 立樹伊波，真治河野：有線 LAN 上の PC 画面配信システム TreeVNC の改良，第 57 回プログラミングシンポジウ

ム予稿集，Vol. 2016, pp. 29–37 (2016).
[5] TightVNC Software: http://www.tightvnc.com.
[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).
[8] 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).