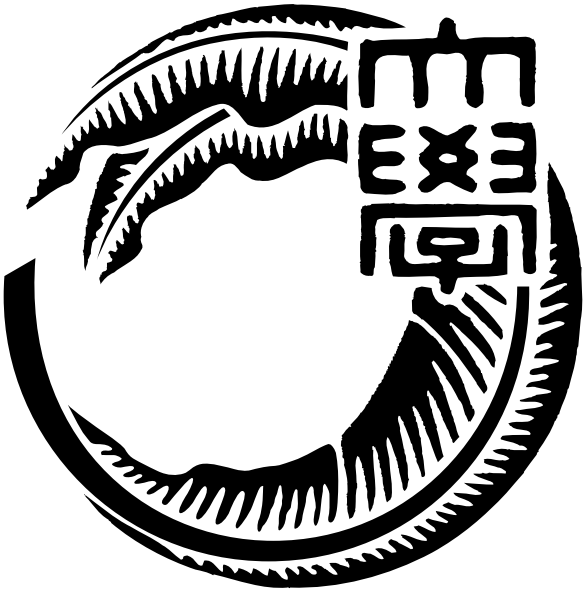
令和元年度 卒業論文

画面配信システム TreeVNC の マルチキャスト導入



琉球大学工学部情報工学科

165729B 安田亮

指導教員 河野 真治

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景と目的

1.2　論文の構成

第2章　TreeVNCの基本概念

2.1　Virtual Network Computing

Virtual Network Computing(以下 VNC) は、サーバ側とクライアント (ビューワー) 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することで可能としている。

2.2　Remote Frame Bufferプロトコル

Remote Frame Buffer(以下 RFB) プロトコルとは VNC 上で使用される、自身の PC 画面をネットワーク上に送信し他人の PC 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、画面送信を行うために 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の確認や認証が行われる。その後、RFB 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 の変更部分を送信する。さらに、RFB クライアントから Framebuffer - UpdateRequest 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、更新があるたび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。

2.3 TreeVNCの構造

TreeVNC は java を用いて作成された Tight VNC を元に作成されている。 TreeVNC は VNC を利用して画面配信を行っているが、従来の VNC では配信 (サーバ) 側の PC に全ての参加者 (クライアント) が接続するため負荷が大きくなってしまう (図 2.1)。

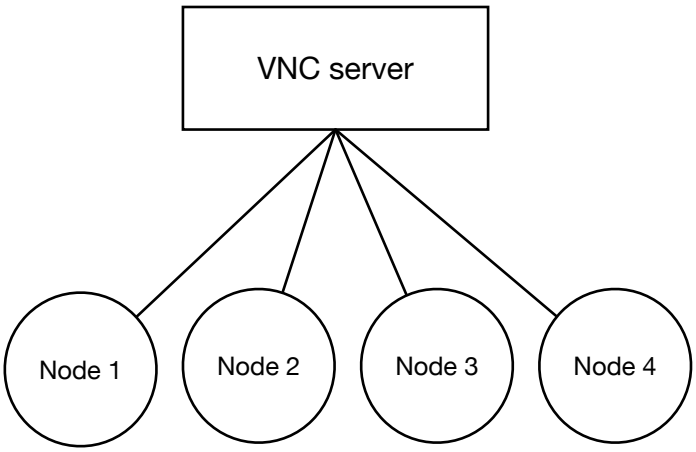


図 2.1: 従来の VNC での接続構造

そこで TreeVNC ではサーバに接続を行ってきたクライアントをバイナリツリー状 (木構造) 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を最大 2 つ接続していく。これにより人数分のデータのコピーと送信の手間を分散することができる (図 2.2)。

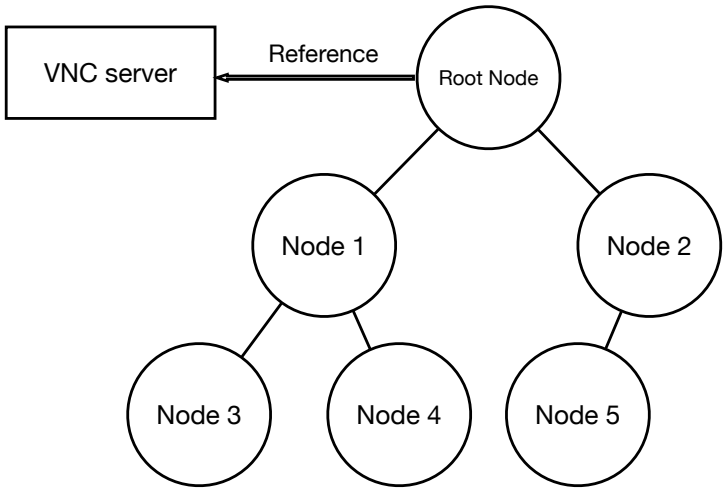


図 2.2: TreeVNC での接続構造

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を行ってきた場合、従来の VNC ではサーバ側が N 回のコピーを行って画面配信する必要があるが、TreeVNC では各ノードが最大 2 回ずつコピーするだけで画面配信が可能となる。

送信されるデータは従来の方法だと N 個のノードに対して N-1 回の通信が必要である。これはバイナリツリー状の構造を持っている TreeVNC で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。

2.4 TreeVNC の通信プロトコル

TreeVNC の通信経路としては以下が挙げられる。

- 任意の Node から Root Node に直接通信を行う send direct message (Node to Root)
- Root Node から任意の Node に直接通信を行う send direct message (Root to Node)
- 任意の 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 から木の末端の Node までの全ての Node に通信を行う message down tree (Root to Node)
- Root Node から配信者の VNC サーバへの通信を行う send message (Root to VNC Server)
- 配信者の VNC サーバから Root Node への通信を行う send message (VNC Server to Root)

Node 間で行われるメッセージ通信には 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 2.1 に示す。

表 2.1: 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.

2.5 MulticastQueue

配信側の画面が更新されると、VNC サーバから画面データが FRAME_BUFFER_UPDATE メッセージとして送らる。その際、画像データの更新を同時に複数の Node に伝えるために MulticastQueue というキューにデータを蓄積している。各 Node はこのキューから画像データを取得し、画面の描画を行う。

2.6 木の再構成

TreeVNC はバイナリツリー状での接続のため、Node が切断されたことを検知できずにいる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 の木構造の接続形態は Root Node が持っている nodeList というリストで管理している。nodeList 内の treeNum という値が各 Node に割り当てられており、これによって木構造のどの位置に Node が接続されているかの判別が可能である。よって、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 切断の検知および木構造の再構成を行っている。

LOST_CHILD の検出方法には、MulticastQueue を用いている。ある一定時間 MulticastQueue から画像データが取得されない場合 MemoryOverFlow を回避するために Timeout スレッドが用意されている。Timeout を検知した際に Node との接続が切れたと判断する。LOST_CHILD の検知と木構造の再構成の手順を以下に示す。

- 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 2.3 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 2.3 中、2: updateNodeList())
- 切断した Node を nodeList から削除し、nodeList の最後尾の Node に切断した Node と treeNum 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 に接続するよう、COMMECT_TO メッセージを送信する (図 2.3 中、3: connectTo(1))
- 最後尾の Node が子 Node を失った親 Node へ接続を行う (図 2.3 中、4: connectToParent(1))

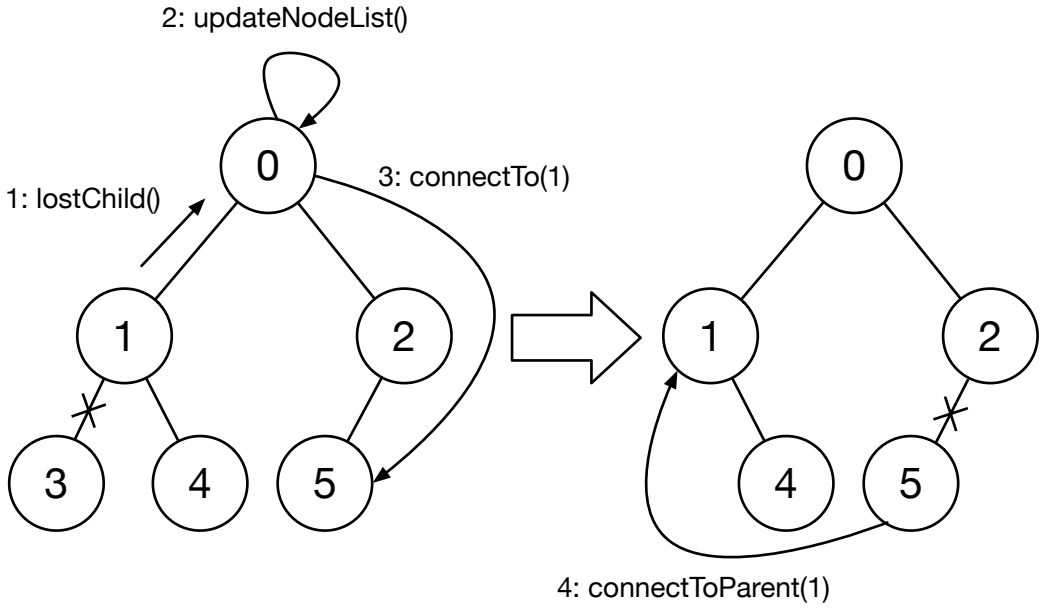


図 2.3: LOST_CHILD の検知・再接続

LOST_CHILD によって、切断された全ての 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、新しく接続を行ってきた Node を適切な場所に接続することが可能となる。

2.7 データの圧縮形式

TreeVNC は ZRLEE というエンコードタイプでデータのやり取りを行う。ZRLEE は RFB プロトコルで使用できる ZLRE というエンコードタイプを元に生成される。

ZLRE 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与され送信される。Zlib は java.util.zip.deflater と java.util.zip.inflater で圧縮と解凍が行える。しかし javaùtil.zip.deflater は解凍に必要な辞書を書きだす (flush) ことが出来ない。従って、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、後述する 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を出来るようにした (2.4)。

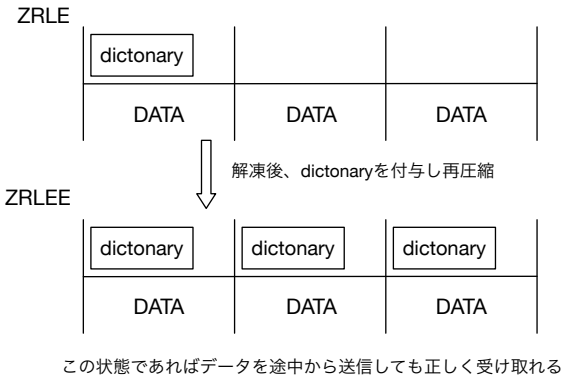


図 2.4: ZRLEE へ再圧縮 f

TreeVNC では RFB プロトコルによって配信側の画面の変更部分は FRAME_BUFFER_UPDATE メッセージとして送られてくる。メッセージの中には変更部分の原点の x,y 座標と縦横の幅が含まれており、長方形として展開される。この長方形を update Rectangle と呼ぶ。以下の表 2.2 に update Rectangle の構成を示す。

2.8 ShareScreen

表 2.2: 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

第3章　実験

3.1　実験説明

3.2

3.3　検証結果

3.4　考察

第4章　今後の課題

参考文献

[1] hogetestusggdgs

謝辞

本研究の遂行，また本論文の作成にあたり、御多忙にも関わらず終始懇切なる御指導と御教授を賜わりました hoge 助教授に深く感謝したします。

また、本研究の遂行及び本論文の作成にあたり、日頃より終始懇切なる御教授と御指導を賜わりました hoge 教授に心より深く感謝致します。

数々の貴重な御助言と細かな御配慮を戴いた hoge 研究室の hoge 氏に深く感謝致します。

また一年間共に研究を行い、暖かな気遣いと励ましをもって支えてくれた hoge 研究室の hoge 君、hoge 君、hoge さん並びに hoge 研究室の hoge、hoge 君、hoge 君、hoge 君、hoge 君に感謝致します。

最後に、有意義な時間を共に過ごした情報工学科の学友、並びに物心両面で支えてくれた両親に深く感謝致します。

2010 年 3 月

hoge