アプリケーション仮想共有における共有管理機構のインタフェース

岩田 知　大囿 忠親　新谷 虎松

我々は以前に WebRTC を用いたアプリケーション仮想共有システムの開発を行ったが，接続方法に関して簡便にアプリケーションの共有ができないという課題があった．本研究では，アプリケーション仮想共有システムにおける共有管理機構インタフェースの開発を行った．本システムではソフトウェアおよびハードウェアを経由し，接続するインタフェースとなっている．これにより，より簡潔な接続を実現した．本稿ではアプリケーション仮想共有システムおよびその共有管理のインタフェースの概要を示す．また，本システムが使用するメモリを測定し，本システムの実用性について評価を行った．

We developed a virtual control mechanism for sharing native applications, but there was a problem about a user interface for link management. In our study, we developed a user interface for link management. An interface of this system to connect via software and hardware. The interface realized a simple and easy connection. In this paper, we show a virtual control mechanism for sharing native applications and a user interface for link management. Moreover, we measure memories used by our system and evaluate practicality of it.

1 はじめに

VNC 等のリモートデスクトップにより他の PC 上で動作するアプリケーションを利用する場合，デスクトップ全体を共有するのではなく，必要なウィンドウのみを共有したい場合がある．ここで，実際にアプリケーションが動作する計算機をホストと呼び，仮想的にアプリケーションを利用する計算機をゲストと呼ぶとする．本研究では，アプリケーション仮想共有システムの開発中である [1]．本研究におけるアプリケーション仮想共有システムとは，ある PC のデスクトップ上に表示されたネイティブアプリケーションのウィンドウを，異なる PC のデスクトップ上に仮想的なウィンドウとして表示する，ウィンドウ単位のリモートデスクトップのようなシステムのことを指す．また，リモートデスクトップのように，表示されたウィンドウに対する操作は元のウィンドウに反映される．これにより，複数の PC 間でアプリケーションを仮想的に共有することが可能になる．

ホスト上の各ウィンドウをどのゲストと共有するのかを指定するための方法を簡単にする必要がある．この時，利便性を高めるために，ウィンドウの共有開始を，アクセス管理を考慮しつつ，ホスト側からもゲスト側からも簡便に実行できるようにしたい．以前に開発したシステムでは，ホストに，異なる PC の Web ブラウザ上から接続し，表示された Web ページにウィンドウを表示することで，共有を行うというインタフェースとなっていた．しかし，この接続インタフェースには使用性に関して課題があった．本システムでは，アプリケーション仮想共有のために，ホストの IP アドレスが必要である．しかし，IP アドレス自体が複雑な数字列であり，暗記や入力が手間である．また，システムを利用する際は，どの IP アドレスがどの PC へ割り当てられたものなのか確認しなければいけないため，システムに関係のない操作が必要で

A User Interface for Link Management on a Virtual Control Mechanism for Sharing Native Applications

Satoru Iwata，Tadachika Ozono，Toramatsu Shintani，名古屋工業大学大学院情報工学専攻, Department of Computer Science, Graduate School of Engineering, Nagoya Institute of Technology.

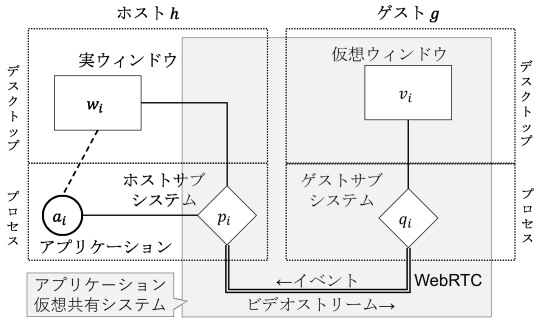

図 1　システム概念図

ある場合が出てきてしまう．

本研究では，先に挙げた課題を，PC に対するラベル付けや，USB メモリを用いることで，ソフトウェアおよびハードウェアを経由した，より簡潔な接続を可能とする接続インタフェースの開発を行った．本稿ではアプリケーション仮想共有システムおよびその共有管理のインタフェースの概要と有効性を示す．

以降の構成を以下に示す．2 章では関連研究について述べる．3 章では，本研究で開発したアプリケーション仮想共有システムの機能および接続インタフェースについて述べる．4 章では，本システムの実装について述べる．5 章では，評価実験および考察について述べる．最後に 6 章で本稿についてまとめる．

2　関連研究

複数の計算機間で連携を行うシステムに関する研究が多くされている．池松ら [2] や米澤ら [3] は計算機間でのファイル共有を直感的に行うためのシステムを開発した．また，萩原ら [4] は iPad からデスクトップアプリケーションのウィンドウを閲覧するシステムを開発した．萩原らの研究は画像処理に焦点を当てたものであり，また，操作などはできず閲覧のみを行うためのシステムであったが，本研究では複数 PC を効率的な利用に焦点を当て，デスクトップアプリケーションのウィンドウを共有するシステムの開発を行った．しかし，以前のシステムは第 1 章で挙げたような課題があり，解決する必要があった．IP アドレスの入力は手間であるため，PC にラベルを付け入力を容易にした．また，その場にいるユーザが全員同じウィンドウを共有するために，それぞれ何かしらの情報を入力することは不効率であると考えられるため，簡潔な動作で共有が行える仕組みが必要である．本研究では USB メモリに焦点を当てた．USB メモリは利便性が高く，安価である．持ち運びも容易であり，近年では学会などで論文データを配布する際などにも用いられる [5]．このように，USB メモリはデータを物理世界でやり取りする際には最適なハードウェアであると考え，本システムでも USB メモリをユーザ間で回し，共有に必要な情報を PC に接続することで取得するようなインタフェースとした．

3　アプリケーション仮想共有システム

本章では本研究で開発したアプリケーション仮想共有システムについて，実行例や図を用いながら述べる．本システムは PC のデスクトップ上に表示されたウィンドウを，異なる PC のデスクトップ上に表示する．図 1 は本システムの概念図を示している．実ウィンドウ w_i は，共有されたアプリケーション a_i のウィンドウを示す．また，仮想ウィンドウ v_i は，その実ウィンドウに対応する仮想的なウィンドウを示す．ホスト h は実ウィンドウを持つ PC を示し，ゲスト g は仮想ウィンドウを持つ PC を示す．p_i と q_i は，本システムのホスト側，ゲスト側で動くそれぞれのシステムを示す．下部の二重線はホストサブシステム p_i とゲストサブシステム q_i が通信していることを示す．また，ここでは WebRTC による Peer-to-Peer 通信を行なっている．

本システムは実ウィンドウを仮想ウィンドウとして表示することができる．また，仮想ウィンドウで発生したイベントを実ウィンドウに反映させることができる．これにより，実ウィンドウを表示している a_i のプロセスに対し，外部の計算機からイベントを送ることができる．これらの機能を実現するために，p_i は，イベントやビデオストリームを Peer-to-Peer 通信で q_i とやり取りする．図 1 ではホストとゲストが異なる PC として示されているが，ホストとゲストを同一の PC としてでも利用することができる．つまり，ゲストとしてあるホストの実ウィンドウを仮想ウィン

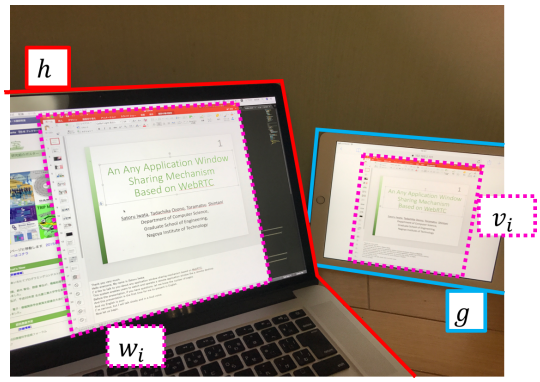

図 2　アプリケーション仮想共有実行例

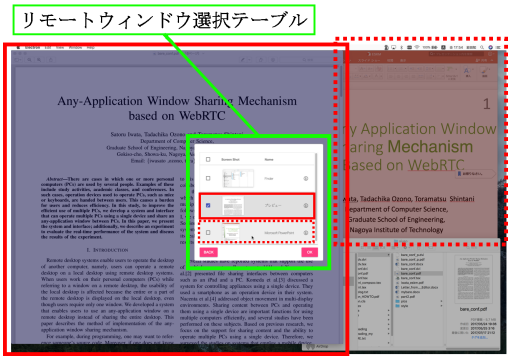

図 3　実ウィンドウ選択画面

ドウとして参照しつつも，ホストとして自分が持つ実ウィンドウを他のゲストへ参照させることもできる．

3.1　実行例

本システムは iPad をゲストとして仮想ウィンドウを表示することもできる．図 2 は iPad をゲストとして本システムを使用した場合の実行例である．実線で囲まれたものが PC h および iPad g であり，点線で囲まれたものがそれぞれ実ウィンドウ w_i およびその実ウィンドウに対応する仮想ウィンドウ v_i である．iPad をゲストとして用いた場合は，マウスやキーボードによる操作ができないため閲覧のみとなってしまうが，タッチ操作によるポインティング機能がある．画面に対しロングタップをすると，実ウィンドウの相対的に同じ座標がポインティングされる．これにより，ゲストが実ウィンドウ上の注目してほしい座標を明確に示すことができる．

3.2　実ウィンドウの選択

ホストは共有を行うために，事前に共有を許可するウィンドウを選択する必要がある．図 3 は実際にウィンドウを選択している場面である．中央の枠で囲まれた部分が実ウィンドウ選択テーブルである．このテーブルでは一つの行が一つのウィンドウと対応しており，行ごとにチェックボックス，サムネイル，アプリケーション名，インフォメーションボタンが表示されている．図 3 では，テーブル内の実線で囲まれた行が，実線で囲まれたウィンドウと対応している．また，点線で囲まれた行は点線で囲まれたウィンドウと対応している．実線で囲まれた行はチェックボックスにチェックが入っているが，これは選択されたことを示す．また，対応するウィンドウは青色にハイライトされる．また，点線で囲まれた行のように，ある行にマウスが乗った場合，その行に対応したウィンドウが赤色にハイライトされ，どのウィンドウを示す行なのかが明確にわかるようになっている．インフォメーションボタンをクリックすると，そのウィンドウの詳細が表示される．パスワードを設定したい場合も詳細から設定することができる．共有を許可するウィンドウの選択が終わったら，OK ボタンもしくは USB メモリを共有デバイスとするボタンを押す．OK ボタンを押した場合は，選択されたウィンドウは一般公開となり，ホストに接続したゲストはそれらのウィンドウから共有したいウィンドウを選択する．もちろんパスワードの設定があるものはパスワードを入力しなければ共有できない．USB メモリを共有デバイスとするボタンを押した場合は，ホストに接続された USB メモリの中から一つ選択し，その USB メモリに共有に必要な情報を書き込む．するとその USB メモリが共有に必要なデバイスとなり，USB メモリを接続することで，選択したウィンドウが共有できるようになる．

3.3　共有管理のインタフェース

本研究で開発した共有管理のインタフェースには，ラベル付けによりソフトウェアを経由するものと，

USB メモリによりハードウェアを経由するものの二つがある．本節ではそれぞれについて述べる．

まずラベル付けを用いた接続インタフェースについて述べる．ラベル付けでは，ホストの PC に対して，誰の PC なのかを示すためのラベルを，ホストおよびゲストが任意に付けることができる．これにより，ただ IP アドレスをやり取りするより，容易に誰の PC なのかを判断できるようになった．ゲストはホスト選択テーブルから，接続先のホストを選択する．必要に応じてパスワードを入力し OK ボタンを押すと，ウィンドウ選択テーブルが表示される．このテーブルにはホストが共有してもよいと設定したネイティブウィンドウが表示される．ゲストはこのテーブルから共有したいウィンドウを選択し，OK ボタンを押すことで仮想ウィンドウが表示される．

次に，USB メモリを用いた接続インタフェースについて述べる．ホストは共有したいウィンドウを選択する．その後，PC に接続した USB メモリを選択し，USB メモリを共有デバイスとする操作を行うと，その USB メモリを共有デバイスとして扱うことができる．ゲストはシステムを起動し，PC に USB メモリを接続するだけで，ホストが選択したウィンドウを共有することができる．

図 4 はホスト選択テーブルを示したものである．テーブルには行ごとにチェックボックスおよびラベルが表示されており，ゲストはチェックボックスにチェックすることでホストを選択することができる．ここでは複数選択が可能である．選択したら OK ボタンをクリックすると，ウィンドウ選択テーブルが表示される．図 5 がウィンドウ選択テーブルを示したものである．テーブルには行ごとにチェックボックス，サムネイル，アプリケーション名，インフォメーションボタンが表示される．こちらも，チェックボックスにチェックすることでウィンドウを選択することができる．また，複数選択も可能である．ウィンドウ選択後は OK ボタンをクリックする．その後選択されたウィンドウが仮想ウィンドウとしてゲストのデスクトップ上に表示され，共有される．iPad 版では，ホスト選択テーブルは表示されるが，複数選択はできなく，またウィンドウ選択テーブルも表示されない．iPad 版ではホストを一つ選択すると，自動でウィンドウ選択テーブル上の一番上に当たるウィンドウが共有対象として選択される．

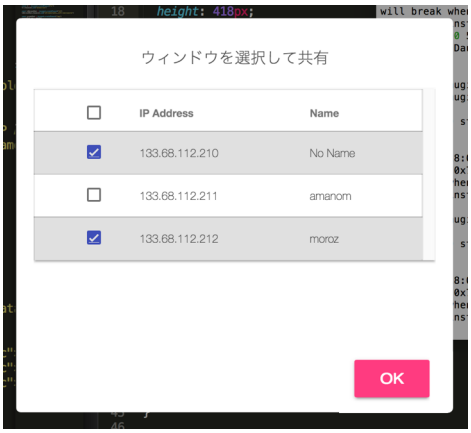

図 4　ホスト選択テーブル

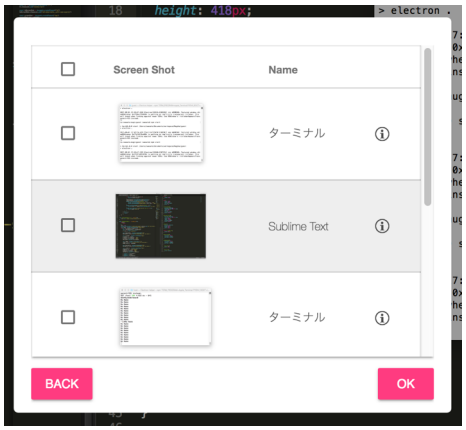

図 5　ウィンドウ選択テーブル

4　実装

本章では本システムの実装について述べる．本システムにはホストサブシステムとゲストサブシステムがあり，どちらもデスクトップアプリケーションとして実装した．また，ゲスト側は Web アプリケーションとしても実装を行ったため，Web ブラウザからシステムを利用することもできる．アプリケーションの実装には Electron を用いることで，Web ブラウザ対応である WebRTC 技術をデスクトップアプリケーションで利用可能とした．ホストとゲスト間の通信に

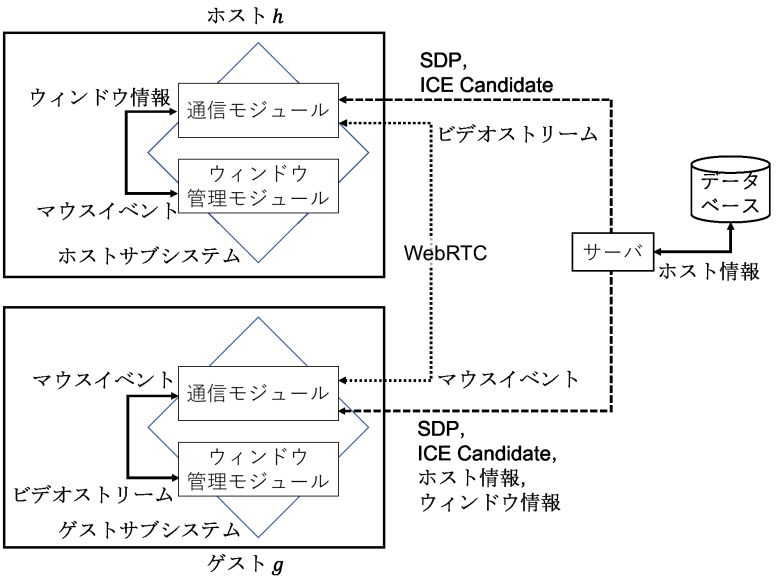

図 6　システム構成

は WebRTC による Peer-to-Peer 通信を用いた．

4.1　システム構成図

図 6 は本システムの構成図を示したものである．本システムはホスト側およびゲスト側でそれぞれホストサブシステム，ゲストサブシステムが動いており，各サブシステム内にはどちらもウィンドウ管理モジュールと通信モジュールが存在する．通信モジュールはシステム間の Peer-to-Peer 通信およびシグナリングサーバを兼ねたサーバとのソケット通信を行う．シグナリングサーバとは，Peer-to-Peero 通信を行うために，必要な情報を Peer 間で交換するためのサーバである．ウィンドウ管理モジュールは文字通りウィンドウを管理する機構であるが，ホストサブシステムとゲストサブシステムで役割が異なる．ホストサブシステム内のウィンドウ管理モジュールでは，実ウィンドウの管理を行う．具体的には，CocoaAPI によりデスクトップ上のウィンドウ情報（アプリケーション名，タイトル，位置，サイズ，ナンバー）を取得する．また，取得したウィンドウ番号およびウィンドウのビデオストリームを取得する．ビデオストリームの取得には JavaScriptAPI の userGetMedia メソッドを用いる．ここで取得したビデオストリームや，ウィンドウに関する情報はまずホストサブシステムの通信モジュールに送られ，その後ゲストサブシステムの通信モジュールへ Peer-to-Peer により送られる．ゲストサブシステムのウィンドウ管理モジュールでは，仮想ウィンドウの管理を行う．具体的には，ホストサブシステムから送られるウィンドウのビデオストリームを仮想ウィンドウで再生する．また，仮想ウィンドウ上で発火したマウスイベントを検知し，検知したイベント情報を通信モジュールへ送る．

サーバではシグナリングサーバとして，Peer-to-Peer 通信に必要な情報をやり取りすると同時に，通信するホストの情報を管理する．ホストの情報には IP アドレスやデスクトップ上の実ウィンドウの情報，ホストに付けられたラベルなどが含まれる．ゲストはサーバから受け取ったホスト情報から，通信可能なホストのラベル一覧を取得する．ゲストに表示されるホスト選択テーブルには，ここで取得したラベル一覧が表示される．

4.2 共有管理機構のインタフェース

アクセス管理を考慮しつつ，ウィンドウの共有管理を，ホスト側からもゲスト側からも柔軟かつ簡便に実行可能にするためのインタフェースを実現した．ここでは，ホスト h 上の実ウィンドウ w をゲスト g 上の仮想ウィンドウ v として，利用可能にすることを共有と呼ぶ．この時，w と v を紐付けることを共有管理と呼び．共有に必要な情報は，ホスト情報およびウィンドウ番号の二つである．以下，これらの情報から共有管理をするための方法を示す．

ホスト情報とは，ホストの SDP や ICE Candidate などの，Peer-to-Peer 通信に必要な情報である．ホスト情報を取得するために，ホストの IP アドレスが必要である．しかし，IP アドレスによる指定は，利便性が悪い．本システムでは，システム内で独自の名前管理を行うことで，ホスト h に割り当てられたラベルにより，ホストを指定可能である．ゲスト上でホスト選択テーブルからホストを選択すると，そのラベルがサーバに送信され，サーバはデータベースを用いてラベルで指定されたホスト h，および h のウィンドウ番号の一覧を取得し，ゲストへ送信する．

ウィンドウ番号とは，ウィンドウ w を一意に特定するための数値であり，w のビデオストリームの取得に利用される．インタフェース上ではゲスト g がウィンドウ選択テーブルからウィンドウを選択した時点で，内部的にウィンドウ番号が取得され，w に対応する仮想ウィンドウ v を生成可能になる．ラベルに基づく名前解決によりホストを特定することで，IP アドレスを用いる場合よりも簡潔な接続が可能になった．

共有管理を簡便化するために，ハードウェアに基づく直感的な共有管理を実現した．ここでは，特別な USB メモリ（以降，ドングルと呼ぶ）を用いた共有管理を実現した．ドングルを用いた共有管理時は，上記の 2 つの情報と ID をドングル内に保存する．具体的には，ホスト上にて，上記の 2 つの情報と ID が，テキストファイルとして，USB メモリに隠しファイルとして保存される．ID は，本システムによって生成された情報であることを識別するための記号列である．その後，ゲストにドングルを接続することで，自動的に共有を開始する．

表 1　実行環境

機種	MackBook Pro
OS	macOS Sierra
プロセッサ	2.5 GHz Intel Core i7
メモリ	16 GB 1600 MHz DDR3
起動ディスク	Macintosh HD
グラフィックス	Intel Iris Pro 1536 MB

ゲストサブシステムは，常にドングルの有無を監視しており，接続されたドングルの妥当性を検証する．具体的には，ID が書かれたテキストファイルが存在するかどうかを調べる．もし存在していた場合は，ウィンドウ共有に必要な情報を取得し，自動でウィンドウ共有を行う．

5 評価

5.1 評価実験

本システムは他のアプリケーションと同時に用いられると考えられる．そのため，本システムが他のアプリケーションにどれだけの負荷を与えるのか評価する必要がある．本研究では，一つのホストが複数のゲストに対しビデオストリームを配信する場合を想定し，ゲストの数が増えるに従い，ホストサブシステムがどれだけのメモリを使用するのかを測定した．使用メモリの値は，Electron の Process オブジェクトが持つ getProcessMemoryInfo メソッドで取得したメモリ使用統計情報の中から，workingSetSize プロパティの値を使用メモリとして取得した．表 1 は評価実験を行った計算機のスペックを示すものである．図 7 が実験結果を示したものである．横軸はホストが通信するゲストの数を表し，縦軸は使用する物理メモリ [メガバイト] を表す．また，今回の実験はウィンドウサイズが 1,920x1,200 であるウィンドウを実ウィンドウとした．

5.2 考察

ゲスト数が 0〜10 の時は，使用する物理メモリはおよそ 100[MB] であり，大きな変化も見られない．ゲスト数が 10 を超えると，使用する物理メモリも大きく増え始め，最終的には 120[MB] を超える値になっ

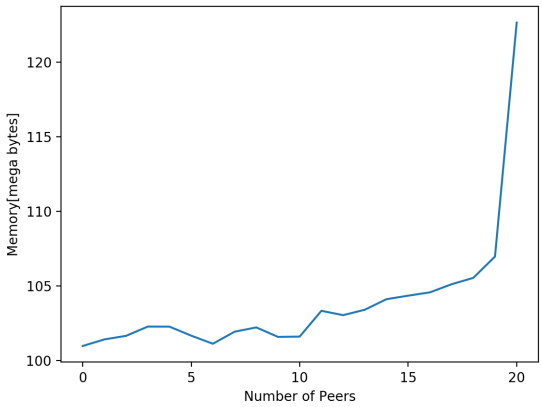

図 7　使用メモリ

た．これらのことから，通信するゲスト数が増えるほど使用する物理メモリは増えていくと思われるが，一つのホストに対し，ゲスト数が 0〜10 であれば比較的安定して共有が可能であることがわかった．

6　おわりに

本稿ではアプリケーション仮想共有システムおよびその共有管理のインタフェースの概要と有効性を示した．アプリケーション仮想共有システムは，ウィンドウ単位のリモートデスクトップを実現し，複数の PC 間でアプリケーションを仮想的に共有することが可能となった．また，共有管理のインタフェースを開発し，以前の接続インタフェースの使用性についての課題を，ホストに対するラベル付けおよび USB メモリを用いた共有管理のインタフェースにより解決した．また，本システムの使用メモリ量を測定し，実用性について評価した．その結果，一つのホストに対し，ゲスト数が 0〜10 であれば安定して共有が可能であることがわかった．本システムにより，WebRTC を用いたリアルタイムなアプリケーション仮想共有が可能になる．例えば情報技術の授業や会議などで応用できると考えられる．

謝辞 本研究の一部は JSPS 科研費 JP15K00422, JP16K00420 の助成を受けたものです．

参 考 文 献

[1] Satoru Iwata, Tadachika Ozono and Toramatsu Shintani: Any-Application Window Sharing Mechanism based on WebRTC, *Proc. 5th International Conference on Smart Computing and Artificial Intelligence*, 2017, pp. 1–6.

[2] 池松香，椎尾一郎: 記憶の石：マルチタッチを利用したデバイス間情報移動, 情報処理学会論文誌, Vol. 55, No. 4(2014), pp. 1344–1352.

[3] 米澤拓郎，中澤仁，永田智大，徳田英幸: Vinteraction: スマート端末のための振動を利用した情報送信インタラクション, 情報処理学会論文誌, Vol. 54, No. 4(2013), pp. 1498–1506.

[4] 萩原拓真，高嶋和毅，モルテンフィールド，北村喜文: モバイルカメラを用いたデバイス間アドホックアプリケーション共有, インタラクション *2017*, pp. 1–9.

[5] 横森励士，青山幹雄，井上克郎: 第 31 回ソフトウェア工学国際会議（ICSE2009）参加報告, 情報処理学会研究報告, No. 2(2009), pp. 1–8.