即応的なネイティブアプリ仮想共有のための共有管理システムの実現

岩田 知[1,a)]　大囿 忠親[1,b)]　新谷 虎松[1,c)]

概要：PC を用いた協調作業では，ネイティブアプリケーションを共有することで作業の効率化が期待できる．本研究では WebRTC 技術を用いて任意のアプリケーションを低遅延で共有することができる，アプリケーション仮想共有システムを開発した．また，共有開始手続きを円滑に行うための，アプリケーション仮想共有ドングルを実現した．ドングルの挿入をキーとして，アプリケーション共有に必要な手続きを自動化することで，迅速な共有開始を可能とした．また，評価実験により，共有開始に要する時間は，最大でも 2,000ms であり，本システムが実用的であることを示した．

Developing a Management System for Instant Native Application Virtual Sharing

Satoru Iwata[1,a)]　Tadachika Ozono[1,b)]　Toramatsu Shintani[1,c)]

Abstract: Sharing native application windows of multiple PCs are demanded to help cooperative works because users can manipulate mixed native windows from different PCs by using their own PCs. However, making composite native applications is time consuming. We have developed an application virtual sharing system with the WebRTC technology. Moreover, we have developed an application virtual sharing dongle for smooth and effective interface. The dongle helps users to share composite native applications easily. The time required to start the application sharing was less than 2,000 ms. This paper shows how the system can be used practically.

1. はじめに

PC を用いた協調作業では，参加者間で作業中の画面を共有することで，効率的な作業が期待できる．例えば，参加者が PC を持参し，システム開発の打ち合わせを行う場面を想定する．このような場面では，ユーザインタフェースの図案などをメンバー全員で共有する際，画面を共有することで，メンバーの意見を参考にしつつリアルタイムで図案を編集し，かつ編集した図案を共有することができる．しかし，アドホックに行われる小規模な協調作業では，作業中の PC 画面を参加者間で共有するための外部ディスプレイが用意できない場合が多い．このような場合，デスクトップ単位で画面を共有するリモートデスクトップや，ネイティブアプリケーションのウィンドウ単位で画面を共有するアプリケーション共有を利用することで，作業中の PC 画面を共有することができる．本研究ではアプリケーション共有について焦点を当てている．アプリケーション共有を行うには，共有するアプリケーションと共有相手を，ユーザが選択する手続き (以下，共有開始手続きと呼称する．) が必要である．この共有開始手続きが円滑に進まない場合，かえって協調作業を妨げてしまう．そのため，協調作業で用いられるアプリケーション共有システムには，円滑な共有開始手続きが行えるインタフェースが必要である．

本研究の目的は，アプリケーション共有システムの開発および，円滑な共有開始手続きを可能とするインタフェー

[1] 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
[a)] iwasato@toralab.org
[b)] ozono@nitech.ac.jp
[c)] tora@nitech.ac.jp

スを実装することにより，協調作業の支援を行うことである．ただし，ここで対象とするのは，四，五人で行われる小規模かつ対面式の協調作業である．本研究では，アプリケーション画面を P2P 通信におけるストリーミング配信することでアプリケーションの共有を行うアプリケーション仮想共有システム [1] を開発した．また，USB メモリの挿入を共有開始のキーとして，共有するアプリケーションおよび共有相手の決定を自動で行うアプリケーション仮想共有ドングルを開発した．また，本研究において，アプリケーションを公開する側のユーザをホスト，公開されたアプリケーションを共有する側をゲストと呼称する．また，公開するアプリケーションを実ウィンドウ，共有されたウィンドウを仮ウィンドウと呼称する．

本稿の構成を以下に示す．第 2 章では関連研究について述べる．第 3 章では，開発したアプリケーション仮想共有システムおよびアプリケーション仮想共有ドングルについて述べる．第 4 章では，本システムの実装について述べる．第 5 章では，本システムの実用性についての評価実験および考察について述べる．最後に第 6 章で本稿についてまとめる．

2. 関連研究

本章では関連研究について述べる．協調作業での利用を想定したアプリケーション共有についての論文は数多く存在する．佐藤ら [2] はアプリケーションのバージョンを考慮した共有システムを開発した．彼らのシステムでは，各参加者が自分の使い慣れたバージョンの操作性をそのまま保持しながら，アプリケーションを共有できる．彼らは個人での利用が想定されたアプリケーションをそのまま複数人で利用できるようなシステムを目指したが，本研究ではあくまでアプリケーションの操作主はホストであり，複数のゲストが共有されたアプリケーションを利用する場面は想定しない．山之上 [3] は P2P 技術を利用したアプリケーション共有システムを開発した．彼らのシステムを組み込んでアプリケーションを開発することで，低遅延なアプリケーション共有が可能となる．本研究でも P2P 技術を利用したアプリケーション共有システムを開発する．加えて，本研究では Web 技術を利用することで，共有のために事前にアプリケーションへシステムを組み込む必要がなく，任意のアプリケーションを共有することができる．

萩原ら [4] はモバイルカメラによる撮影をキーとして共有開始手続きを完了するシステムを開発した．モバイルカメラは携帯端末であり持ち運びが容易であるため，アドホックな状況における円滑なデバイス間でのデータ共有に応用できる．本研究では，アプリケーション共有をホスト，ゲストともに PC 上で行うことを想定している．そのため，計算機に搭載された，もしくは付属のカメラなどを用いたインタフェースは本システムには不適であるが，持ち運びが容易なハードウェアをインタフェースに応用することで，共有開始手続きが円滑になる可能性がある．以上のことを踏まえ，本研究では共有開始手続きのためのインタフェースにはドングルとして USB メモリを用いた．USB メモリは安価で入手が容易である．また，小型な記憶媒体であるため持ち運びも容易である．事前に共有に必要な情報を選択しておき，USB メモリの挿入をキーとして共有開始手続きが自動で完了するようなインタフェースとすることで，円滑な共有開始手続きを可能にする．

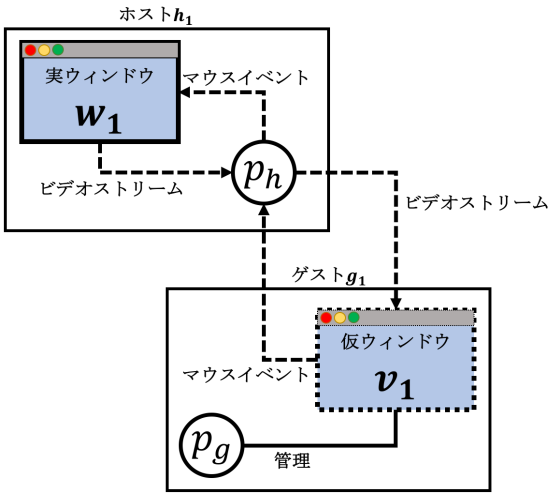

図 1 概念図
Fig. 1 Conceptual Diagram.

3. アプリケーション共有システム

本章では，実行例や図を用いながら，本研究で開発したアプリケーション共有システムの機能やインタフェースについて述べる．第 1 節ではシステム全体の説明を述べる．次に，共有開始手続きにおけるインタフェースについて述べる．共有開始手続きにおけるインタフェースには，ソフトウェアベースのものと，ドングルを用いたアプリケーション仮想共有ドングルの二つが存在する．第 2 節でソフトウェアベースのインタフェースについて述べ，第 3 節でアプリケーション仮想共有ドングルについて述べる．

3.1 アプリケーション仮想共有システム

本システムはホスト側のデスクトップ上に表示されたアプリケーションのウィンドウを，ゲスト側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．ウィンドウ w_1 は，ホスト h_1 のデスクトップ上に存在する実ウィンドウを示す．また，ウィンドウ v_1 は，実ウィンドウ w_1 に対応した，ゲスト g_1 のデスクトップ上に存在する仮想的なウィンドウを示す．プロセス p_h はホスト h_1 上で動くプロセスを示し，プロセス p_g はゲスト g_1 上で動くプロセスである．プロセス p_h は，実ウィンドウ w_1 からビデオストリームを取得する．また，プロセス p_h が受

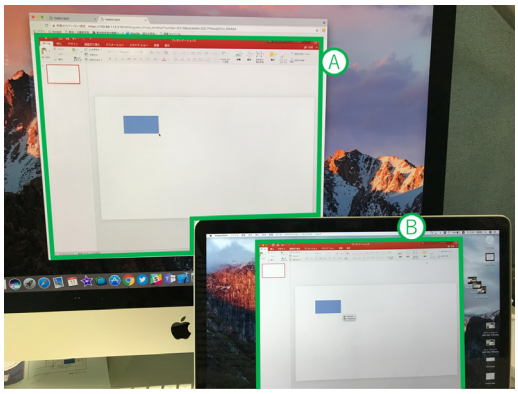

図 2 実行例：アプリケーション共有
Fig. 2 Example: Application Sharing

信したマウスイベントを実ウィンドウ w_1 へポストすることである．ここで取得したビデオストリームは仮ウィンドウ v_1 へ送信される．仮ウィンドウ v_1 は受け取ったビデオストリームをウィンドウ上で再生し，画面共有を行う．また，仮ウィンドウ v_1 はマウスイベントを検知し，検知したマウスイベントをプロセス p_h へ送信する．プロセス p_g は，仮ウィンドウ v_1 の作成および削除を行う．

図 2 は本システムを利用した場合の実行例である．実線で囲まれたウィンドウ A は，ゲスト側の PC のデスクトップ上に存在する共有されたアプリケーションのウィンドウを示す．実線で囲まれたウィンドウ B は，ホスト側の PC のデスクトップ上に存在する公開されたアプリケーションのウィンドウを示す．図のように，ウィンドウ B と同じ画面がウィンドウ A に表示されている．また，ウィンドウ A に行ったマウスダウン，マウスアップなどの操作はウィンドウ B に反映される．

3.2 ソフトウェアベースのインタフェース

本節ではソフトウェアベースのインタフェースについて述べる．本システムのインタフェースは，ホストが公開するアプリケーションを選択するインタフェースと，ゲストが共有するアプリケーションを選択するインタフェースの二つに分割することができる．まずはホスト側のインタフェースについて述べ，次にゲスト側のアプリケーションについて述べる．

ホストには，リスト形式で PC 上で起動するアプリケーションのウィンドウのスクリーンショットやアプリケーション名などの情報を提示する．図 3 は実際に公開するアプリケーションをホストが選択している場合のデスクトップ画面である．中央に表示されている実線で囲まれたウィンドウは実ウィンドウ選択リストである．リスト中の実線で囲まれた行，および点線で囲まれた行は，それぞれデスクトップ上の実線，および点線で囲まれたウィンドウと対応している．デスクトップ上の実線で囲まれたウィンドウは青くハイライトされており，点線で囲まれたウィンドウ

図 3 実行例：公開するアプリケーションの選択
Fig. 3 Example: Selecting Application in Host Side.

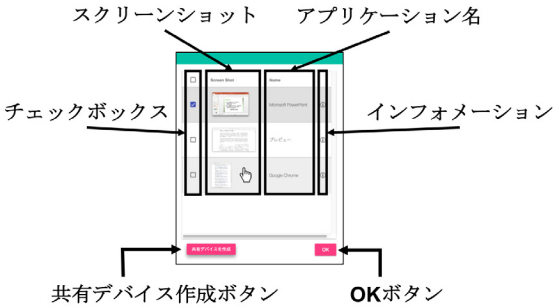

図 4 実ウィンドウ選択リスト
Fig. 4 Application List for Selecting.

は赤くハイライトされている．青いハイライトは，選択されたウィンドウを示す．また，赤いハイライトは対応する行にマウスオーバーしていることを示す．

図 4 は図 3 上の実ウィンドウ選択リストを拡大したものである．各行には，左から順にチェックボックス，スクリーンショット，アプリケーション名，インフォメーションアイコンが表示されている．チェックボックスにチェックを入れることで公開するウィンドウを選択できる．インフォメーションアイコンは，クリックすると，対応するウィンドウの詳細情報が表示される．この詳細情報には，ウィンドウサイズや位置などが含まれている．実ウィンドウ選択リスト下部には共有デバイス作成ボタンおよび OK ボタンが表示されている．OK ボタンは選択したウィンドウを決定するボタンである．共有デバイス作成ボタンは，アプリケーション仮想共有ドングルを作成するボタンである．

以上がホスト側のインタフェースになる．次に，ゲスト側のインタフェースについて述べる．ゲスト側では，共有するアプリケーションを選択する前に，ホストの選択を行う．

図 5 はホスト選択リストである．リストにはアプリケーションを公開しているホストの一覧が表示されている．各行には左から順に，チェックボックス，IP アドレスおよびホスト名が表示されている．チェックボックスにチェックを入れることでホストを選択したことになる．ホスト名は，各ホストが任意に設定することができる．ホスト選択

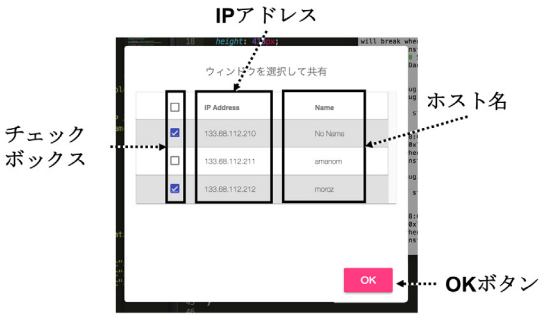

図 5 ホスト選択リスト
Fig. 5 Host List for Selecting.

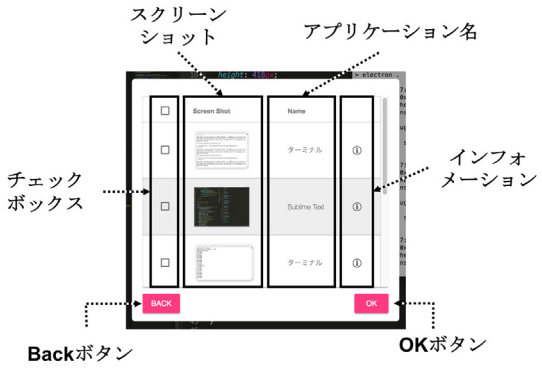

図 6 仮ウィンドウ選択リスト
Fig. 6 Virtual Window List for Selecting.

リスト下部には OK ボタンが表示されている．OK ボタンをクリックすると，次は仮ウィンドウ選択リストが表示される．

図 6 は仮ウィンドウ選択リストを示す．リストには，ホスト選択リストで選択したホストが公開しているアプリケーションのウィンドウの一覧が表示されている．各行には，左から順に，チェックボックス，仮ウィンドウのスクリーンショット，アプリケーション名，インフォメーションアイコンが表示されている．各項目の役割は実ウィンドウ選択リストと同じである．仮ウィンドウ選択リスト下部には Back ボタンおよび OK ボタンが表示されており，Back ボタンを押すと再びホスト選択リストが表示される．OK ボタンを押すと，選択した仮ウィンドウがゲスト側のデスクトップ上に表示される．

3.3 アプリケーション仮想共有ドングル

本節ではアプリケーション仮想共有ドングルについて述べる．まずホストは，先に述べた実ウィンドウ選択リストにより，公開するアプリケーションを選択する．選択を終えたら，共有デバイス作成ボタンを押す．すると，記憶媒体を選択するダイアログが表示され，任意の記憶媒体を選択する．ここで選択された記憶媒体がドングルとなる．次に，選択した記憶媒体をゲスト側の PC に挿入する．すると，ホスト側で選択されたアプリケーションが自動で選択

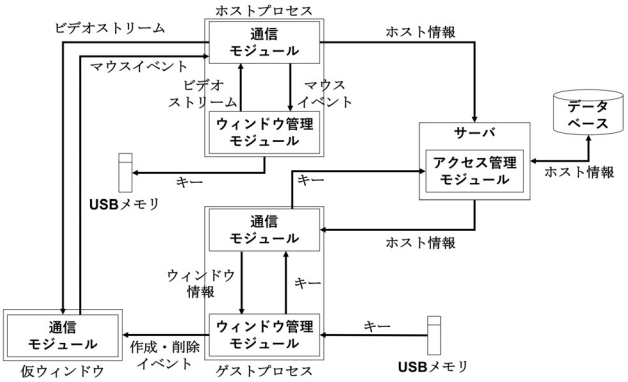

図 7 システム構成図
Fig. 7 System Architecture.

され，共有開始手続きが完了する．ソフトウェアベースのインタフェースでは，ゲストはリストからホストやアプリケーションを選択する必要があるが，アプリケーション仮想共有ドングルはホストが公開したアプリケーションをそのまま共有するため，ゲストが選択する手間を減らすことができる．また，ドングルを用いることでパスワードなどを設定しなくてもセキュアなアプリケーション共有が可能となる．

4. 実装

本章では本システムおよびインタフェースの実装手法について述べる．図 7 は本システムの構成図を示す．本システムは大きくホストプロセス，ゲストプロセス，サーバ，仮ウィンドウの四つに分けることができる．また，図中にある USB メモリが，アプリケーション仮想共有ドングルで利用するドングルである．

ホストプロセスは通信モジュールおよびウィンドウ管理モジュールを持っている．ウィンドウ管理モジュールは，ホスト側のデスクトップ上に表示されているウィンドウの管理を行う．ウィンドウ管理モジュールの役割は，ウィンドウのビデオストリームの取得およびマウスイベントをポストすることである．また，共有に必要な情報を USB メモリへ格納する．通信モジュールはサーバや仮ウィンドウとの通信を行う．サーバには，ホスト側のデスクトップ上に表示されているウィンドウや，それらを一般公開するかどうかなどの情報を送信する．仮ウィンドウには，ウィンドウ管理モジュールで取得したビデオストリームを送信している．また，仮ウィンドウの通信モジュールとは P2P 通信を行なっている．

サーバでは，ホストプロセスおよびゲストプロセス間の通信を行なっている．ここでの通信はソケット通信である．ホストプロセスから送られたホスト情報はデータベースに格納される．ここで格納されたホスト情報は，ゲストプロセスからリクエストがあった際はゲストプロセスへ送信する．また，サーバ内にはアクセス管理モジュールがある．

この機構は，ゲストがホストとアプリケーション共有を行う際に，そのホストが通信可能か，ウィンドウを共有可能かなどを判断し，可能であった場合にゲストに対しウィンドウ情報や P2P 通信を行うために必要な情報を送信する．また，アクセス管理モジュールでは，各ホストが一般公開するウィンドウ群と，各ホストが USB メモリを用いて公開するウィンドウ群をそれぞれグループ化し管理している．各グループにはユニーク ID が割り振られており，ゲストが共有を開始するには，サーバに対しこのユニーク ID をキーとしてアクセスする必要がある．

ゲストプロセスは通信モジュールおよびウィンドウ管理モジュールを内包する．通信モジュールはサーバとの通信を行う．サーバから受け取ったホスト情報から，ホスト名やウィンドウ情報などを抜き出し，ホスト選択リストや仮ウィンドウ選択リストに表示している．また，P2P 通信に必要な情報もホスト情報中のウィンドウ情報に含まれている．ウィンドウ管理モジュールでは通信モジュールから受け取ったウィンドウ情報から仮ウィンドウ作成イベントを発生させ，仮ウィンドウを表示する．また，挿入された USB メモリからキーを共有開始に必要な情報として読み取り，通信モジュールを経由してサーバへアクセスする．仮ウィンドウは作成時にホストプロセスとの P2P 通信に必要な情報を受け取り，仮ウィンドウが表示されたと同時にホストプロセスとの接続を試みる．接続が完了したら，ウィンドウ情報に従いビデオストリームをホストプロセスに要求する．また，仮ウィンドウで検出したマウスイベントをホストプロセスへ送信する．また，P2P 通信やストリーミングの配信には WebRTC 技術を用いた．

5. 評価実験

本章では評価実験について述べる．評価実験では，ゲストが共有する仮ウィンドウを決定してから，その仮ウィンドウが表示されるまでの経過時間の計測およびその内訳を調査する．また，その結果について考察し，本システムの実用性について評価する．

5.1 計測手法

まずは全体の時間の計測手法について説明する．実際に仮ウィンドウ画面が表示されたかどうかを判断するために，ゲスト側のデスクトップをビデオカメラで撮影する．また，ネットワークの環境による影響を抑えるために，サーバ，ホストプロセスおよびゲストプロセスは全て同じ計算機上で実行する．以降，仮ウィンドウが表示されるまでの経過時間の計測手順について説明する．

まず，ホストは一つのアプリケーションを公開し，ゲストはその公開されたウィンドウ画面を共有する．一定時間ごとにゲストは同じアプリケーションのウィンドウ画面を，新しい仮ウィンドウとして表示する．仮ウィンドウは30 秒ごとに 1 枚ずつ増やしていき，10 枚に達したところで共有を終了する．これらの様子をビデオカメラにより撮影する．撮影後，共有開始の操作を完了してから，決定した仮ウィンドウ画面が表示されるまでのフレーム数をカウントする．カウントしたフレーム数をビデオカメラの FPS で割り，仮ウィンドウ画面が表示されるまでに要した時間を計測する．また，今回は FPS が 60 のビデオカメラを用い，共有するウィンドウ画面のサイズは 960x1,200 とした．

表 1 計測項目
Table 1 Measurement Items.

エンコード処理時間 [ms]
RTT[ms]
デコード処理時間 [ms]

表 2 用いた計算機
Table 2 Computer Used for Experiment.

OS	macOS Sierra
プロセッサ	2.5 GHz Intel Core i7
メモリ	16 GB 1600 MHz DDR3
グラフィックス	Intel Iris Pro 1536 MB

次に，内訳の取得方法を説明する．本システムは仮ウィンドウのフレームが完全に表示されてからホストと P2P 通信を行うようになっているので，上記の全体時間の計測時に，共有開始の操作を完了してから仮ウィンドウのフレームが完全に表示されるまでのフレーム数もカウントする．このフレーム数を同様にすることで，P2P 通信が始まるまでに要する時間を計測する．また，P2P 通信が始まってからの内訳には chrome://webrtc-internals/を利用する．これは WebRTC における送受信について，P2P 通信が始まってからの動画のエンコードやデコードに要した時間などの統計データを，1 秒間隔で確認することができるデバッグツールである．このツールは上記の全体時間の計測時に同時に利用する．また，このツールを用いて計測する項目について，表 2 に示す．また，本実験では P2P 通信が始まってからの内訳として，仮ウィンドウ画面が表示されるまでのエンコード処理時間やデコード処理時間の総和を取得する必要がある．そのために，フレーム数をカウントすることで取得した全体の時間と，P2P 通信が始まるまでの時間の差から，P2P 通信が始まってから仮ウィンドウ画面が表示されるまでのおよその時間を取得する．この結果から，エンコード処理時間などの総和を求める．表**??**は，評価実験に用いた計算機のスペックを示すものである．以上の手順および条件に従い，各時間を 13 回計測し，それぞれの平均値を取得する．

5.2 実験結果

図 8 は，ビデオカメラで撮影した結果から取得した各時間である．横軸は仮ウィンドウ数，縦軸が計測した時間を

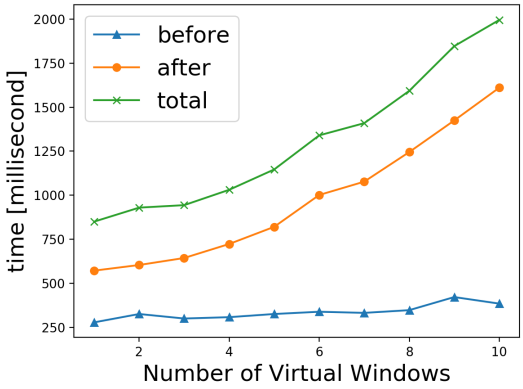

図 8 全体時間
Fig. 8 Total Time.

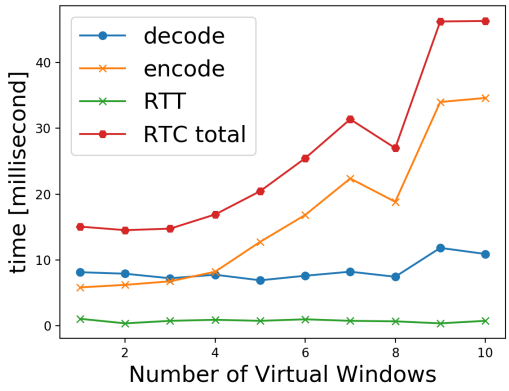

図 9 P2P 通信開始後の処理時間内訳
Fig. 9 Required Time for Processing.

示す．total が全体の時間，before が仮ウィンドウのフレームが表示されるまでの時間，after が P2P 通信が始まってから仮ウィンドウ画面が表示されるまでの時間を示す．仮ウィンドウが増えるに従い，全体時間が大きくなっていることがわかる．また，P2P 通信が始まってからの処理時間が仮ウィンドウ数に依存しており，それまでの時間はほぼ一定であることがわかる．

図 8 から，仮ウィンドウ数が 6 枚の時点で after がほぼ 1,000ms に達し，10 枚では 1,500ms に達した．chrome://webrtc-internals/では 1,000s 間隔でのみ計測時間が提供されているので，少数第一位を切り上げた値の時間だけ，エンコード処理時間などの総和を求めた．図 9 は chrome://webrtc-internals/を用いて計測した P2P 通信開始後の内訳である．横軸は仮ウィンドウ数，縦軸が計測した時間を示す．decode はデコード処理時間，encode はエンコード処理時間，RTT は Round Trip Time，RTC total は全体の総和を示す．decode および RTT は全体を通してほぼ一定であったが，encode は仮ウィンドウ数が増えるにつれて値が大きくなった．

5.3 考察

図 9 より，エンコード処理時間が大きく仮ウィンドウ数に依存していることがわかる．一方で，デコード時間や RTT は仮ウィンドウ数にほぼ依存しないことがわかった．現状のシステムでは，ホストは同プロセス上で複数の通信や動画の配信を行っている．一方，ゲストはたとえ同じデスクトップ上で複数の仮ウィンドウを表示していても，各仮ウィンドウごとに異なるプロセスで通信や動画の受信を行なっている．これにより，ホストでは同プロセスにエンコードの処理が集中したため，ゲストに比べ，通信を行うピア数の増加による影響を顕著に受けたのだと考えられる．

図 8 では，after の値は最小でもおよそ 600ms である．仮ウィンドウ数が増えるにつれ要する時間も増え，10 枚でおよそ 2,000ms となる．本研究で対象とする協調作業は参加者が四，五人ほどであるので，一人のホストが各ゲストに対し 2 枚の仮ウィンドウ画面を共有するとしても，各ゲストは共有開始の操作を行ってから 2,000ms 以内で共有が開始されることになる．この結果に対して，本研究では実用的であると考えている．

6. おわりに

本稿では円滑な協調作業を支援するための，アプリケーション仮想共有システムおよび共有開始手続きを行うためのインタフェースを開発した．ただし，本研究では少人数かつ対面式の協調作業を対象としている．アプリケーション仮想共有システムでは，Web 技術と P2P 通信を用いることで低遅延なアプリケーション共有を実現した．また，アプリケーション仮想共有ドングルを実現することで，インタフェースにおいて共有開始手続きの自動化を可能とした．共有開始に要する時間を計測し，実用性についての評価実験および考察を行った．結果から，少人数の協調作業においてであれば，本システムの共有開始に要する時間は最大でも 2,000ms であり，実用的であると思われる．

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. in Society for Cardiovascular Angiography and Interventions. pp.1-6, 2017.
[2] 佐藤慶三，村上峻吾，谷口智明，中島誠．“アプリケーションの共有による協調作業を促進する異なるバージョン間での操作互換”．電子情報通信学会論文誌．Vol.J97-D, No.8, pp.1307-1317.
[3] 山之上卓．“P2P 技術を利用した分散システム上の実時間操作共有システム”．情報処理学会論文誌，Vol.46, No.2, pp.392-402, 2004.
[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.1-9，2017.