CJ-002

WebRTC SFU に基づくウィンドウ画面共有システムの実装
Implementing a System for Sharing Window Information Using WebRTC SFU

岩田知 †　大囿忠親 †　新谷虎松 †
Satoru Iwata　Tadachika Ozono　Toramatsu Shintani

1. はじめに

協調作業では参加者個人の PC 画面を共有することによって作業の効率化が期待できる．特にウィンドウ単位での画面共有は，デスクトップ単位での画面共有と比較すると必要な情報のみを共有することができる点でより有用である．例えば，他者の画面を閲覧するユーザは，自分のデスクトップ上における作業領域および閲覧性を確保することができる．しかしその一方で，どのウィンドウを誰と共有するのかといった共有の開始に必要な手続きが煩雑になり，管理も困難になってしまう．また，リアルタイムな画面共有には Peer-to-Peer のストリーム配信が有効である．しかし，複数のウィンドウ画面を複数人に配信するウィンドウ単位での画面共有では，共有する端末が増加することで，配信する端末に高い負荷がかかってしまう．

本研究ではウィンドウ単位での複数の画面共有におけるこれらの課題を解決することを目的として，グループ単位で画面共有を行うためのグループ共有管理インタフェースおよびストリームのブロードキャストをサーバに行わせる WebRTC SFU(Web Real-Time Communication Selective Forwarding Unit) の導入を行った．本稿では開発したグループ共有管理インタフェースおよび WebRTC SFU 導入に伴い開発したストリーム管理機構について述べる．また，本システムを用いて画面配信を行った際に使用したメモリ量を，配信する端末の負荷として計測することで，WebRTC SFU 導入の効果について評価した．以下，ウィンドウ単位での画面共有をウィンドウ共有，配信する端末を配信端末，閲覧する端末を閲覧端末と呼称する．

2. 関連研究

本章では関連研究について述べる．協調作業支援を目的とした画面共有システムに関する研究は多く行われている．Buasri ら [1] は仮想授業に導入するための画面共有システムを開発した．画面共有以外にも，ホワイトボードやクイズといった機能を搭載することで，授業を支援した．また，Xue ら [2] は画面共有をビデオ会議に導入した．これらのシステムはどちらも Web ベースで開発されており，WebRTC を利用することで，Web ブラウザ間での P2P 通信を行なっている．WebRTC は Web ブラウザ間で特定のプラグインがなくても通信できる API で，リアルタイムコミュニケーションを支援する．Masaki ら [3] は PC とタブレット間での協調作業支援を行うための画面共有システムを開発した．タブレットは PC 画面の一部分を閲覧することができる．このシステムにより PC 上で行なっている作業について，タブレットを通し遠隔地の人に教えることができる．彼らは PC のスクリーンショットを一定間隔でタブレットに送信することでシステムを実装した．画面配信の手法としては，大きく動画を配信するかスクリーンショットを送信するかの二つが見られた．スクリーンショットを送信する手法は，配信端末への負荷が大きくなる恐れがある．Masaki らのシステムは 1 対 1 で共有する場合が想定されているが，本システムでは多数の端末に対し送信することになる．そのため本システムにおいては送信されるデータの質ではなく，端末にかかる負荷や遅延を考慮する必要がある．また，萩原ら [4] はデスクトップ単位ではなくアプリケーションウィンドウ単位で画面を共有するシステムを開発した．彼らのシステムは PC 画面上のアプリケーションウィンドウをタブレットで撮影することで，PC とタブレット間で撮影したアプリケーションウィンドウ画面を共有することができる．彼らは撮影しそのまま撮影した端末上で共有した画面を閲覧できるというインタフェースにすることによりアドホックな画面共有を可能にした．本研究では PC 間での共有を想定しており，Web カメラや内蔵カメラといったハードウェアを使用すると，かえって共有に必要な操作が煩雑になってしまう恐れがある．

以上のことを踏まえ，我々は先行研究 [5] として，PC 間で不特定なアプリケーションウィンドウ画面を共有することができる仮想ウィンドウ共有機構を開発した．また，画面配信にはリアルタイム性を考慮して WebRTC を用いた．また，インタフェースとしては特別なハードウェアは使用せずソフトウェアベースとした．この先行研究で開発したシステムをベースに，本研究の提案手法であるグループ共有管理インタフェースおよび WebRTC SFU 導入を行なった．

3. 画面共有システム

本章では本研究で開発した画面共有システムの実装およびインタフェースについて述べる．まず，先行研究で開発した仮想ウィンドウ共有について，WebRTC SFU と合わせて述べる．これは本システムにおける配信されたウィンドウ画面のビデオストリームを閲覧するためのインタフェースである．次に，本システムの画面配信部分の構成図を示す．次に，ストリーム管理機構について述べる．最後に，グループ共有管理インタフェースについて述べる．

3.1 仮想ウィンドウ共有

本節では配信されたウィンドウ画面のビデオストリームを閲覧するためのインタフェースである仮想ウィンドウ共有および WebRTC SFU について説明する．図 1 は仮想ウィンドウ共有の図説である．図中には配信端末および閲覧端末が 1 台ずつ存在する．また，配信端末の画面上にはウィンドウが一つ表示されている．これを元ウィンドウとする．図では，これを閲覧端末へ配信している．まず，本システムが元ウィ

† 名古屋工業大学大学院情報工学専攻

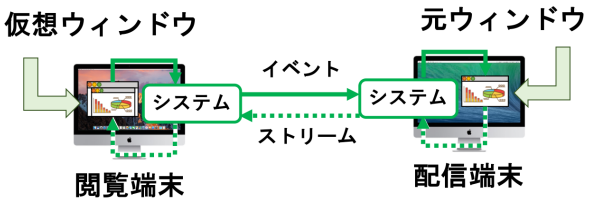

図 1: 仮想ウィンドウ共有

ンドウ画面のビデオストリームを取得する．次に，本システムは取得したビデオストリームを閲覧端末へ送信する．この際，サーバを経由する．閲覧端末は，ビデオストリームを受け取ったらまずは空のウィンドウを作成する．次に，空のウィンドウ上で受け取ったビデオストリームを再生する．これにより，元ウィンドウ画面の配信が可能になる．また，作成したウィンドウ上で検知したマウスおよびキーイベントは，サーバを経由して配信端末に送信される．配信端末はイベントを受け取ったら，元ウィンドウ上で同イベントを発火させる．これにより，遠隔操作が可能になる．

次に，WebRTC SFU について説明する．WebRTC SFU は，本来 P2P で通信を行う WebRTC に，サーバ・クライアント方式を導入したものである．WebRTC で発生する配信端末の高負荷の原因は，閲覧端末が増えるに従い，配信に必要な通信や動画の処理も増えるためである．WebRTC SFU では，ストリームのブロードキャストをサーバが行うことで，閲覧端末の増加に伴う配信端末での負荷を軽減させることができる．図 1 のように，配信端末はストリームをサーバを経由して閲覧端末へ送信する．ストリームを配信端末はサーバととのみ通信を行うため，閲覧端末への依存が少ないシステム設計が可能になる．本システムでは API など利用せず「mediasoup[1]」というライブラリを用いてサーバを構築し，WebRTC SFU の導入を行った．

3.2 システム構成

図 2 は本システムの画面配信部分の構成図である．データベースではクライアントのアカウント情報や作成したグループに関する情報を管理している．クライアントは所属するグループごとに，サーバとのコネクションを内部システムで管理しており，ストリームの配信を行う際はクライアントが指定したグループに対応するコネクションが使用される．つまりクライアントは所属するグループごとにコネクションを持っているということになる．図中にはグループは一つしか示されていないが，実際は複数のグループが存在することになる．このグループについては本章 3 節で詳しく述べる．これによりこの後ブロードキャストするグループを決定している．サーバはクライアントと P2P 通信およびソケット通信を行う．P2P 通信はストリームの配信および受信に用い，クライアントが増加した，グループが作成されたといったイベントはソケット通信によって他クライアントへ通知される．

[1]https://mediasoup.org/

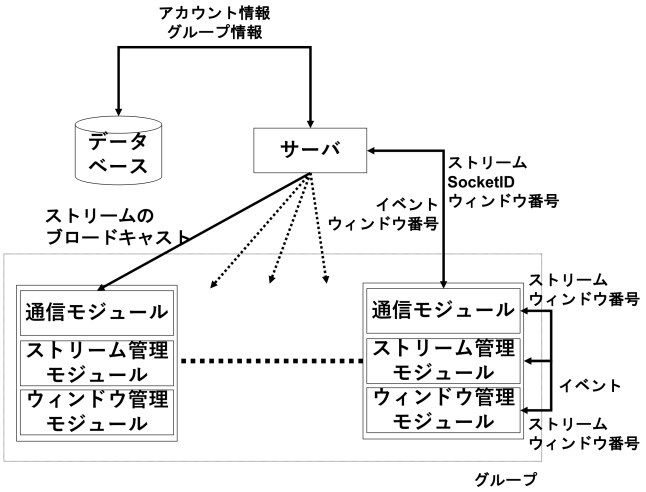

図 2: システム構成図

サーバはクライアントから受け取ったストリームとそれに関する情報として SocketID および Window 番号を同グループに所属するクライアントにブロードキャストする．

クライアントの内部システムでは通信モジュール，ストリーム管理モジュールおよびウィンドウ管理モジュールが存在する．通信モジュールはサーバとの通信を行うための機構である．データの送受信を担当し，受信したデータはイベント情報を除いてストリーム管理モジュールに送る．ストリーム管理モジュールは配信するストリームや受信したストリームについて ID を付与して管理する．ID にはサーバとの通信で利用しているソケットの ID およびウィンドウを一意に特定するためのウィンドウ番号を利用している．このウィンドウ番号は OS がウィンドウ一つ一つに割り当てるもので，本システムが管理しているものではない．ウィンドウ管理モジュールはウィンドウ画面のストリームの取得やウィンドウに対しイベントを送信する役割を持つ．取得したウィンドウ画面のストリームはストリーム管理モジュールを経由して通信モジュールに送られる．

3.3 ストリーム管理機構

WebRTC SFU の導入にあたり，ストリーム配信の最適化と管理が必要になった．本システムは，他のアプリケーションと並行して利用されると想定される．そのため，他のアプリケーションを快適に利用するためにも，本システムを最適化することは重要である．本システムはウィンドウ上でウィンドウ画面のビデオストリームを再生することで仮想ウィンドウ共有を行なっているが，閲覧端末上でウィンドウを閉じることで，その閲覧端末と配信端末間の共有を終了することができる．ここで，閲覧端末の数が 0 になった場合は，配信端末内で保持しているストリームを破棄することで，使用メモリ量を無駄に消費することを防ぐことができる．また，例えば閲覧端末が配信端末から同じビデオストリームが送られ

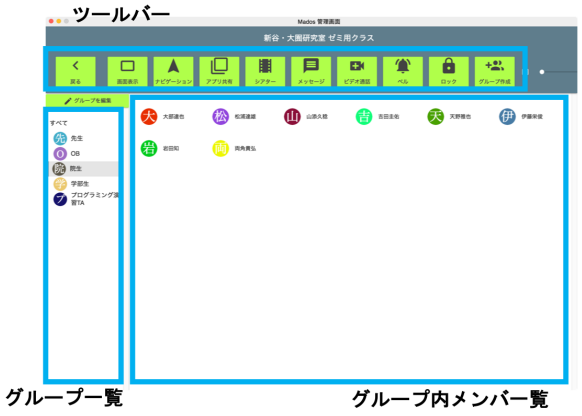

図 3: グループでのウィンドウ共有管理画面

ていた場合は，新しく送信されたビデオストリームを廃棄し，以前に共有が完了している仮想ウィンドウを表示することで，同じく使用メモリ量の消費を抑えることができる．これらのように，本システムの使用メモリ量を最適化するための機構がストリーム管理機構である．ストリーム管理機構は内部でウィンドウ番号というウィンドウ画面を一意に特定するための ID，および誰が送信したウィンドウ画面であるかという情報をソケット ID などから取得し，自分がどのウィンドウ画面を誰に配信しているのか，誰のどのウィンドウ画面を閲覧しているのかといったストリームの送受信の関係を管理している．

3.4 グループ共有管理インタフェース

本節では本システムのグループ共有管理インタフェースについて述べる．本システムではウィンドウ共有を行うユーザの集合 U_i を，ユーザが作成したグループ g_i として管理している．ユーザは事前に本システムを利用するユーザの集合 U から任意にユーザを複数選択し，グループを作成する．

図 3 は本システムにおけるウィンドウ共有の管理画面である．管理画面はツールバー，グループ一覧およびグループ内メンバ一覧の表示部の三つに分かれる．ツールバーでは協調作業支援のための機能を使用するためのボタンが表示されている．ウィンドウ共有機能やグループ作成機能もここに含まれる．また，メッセージング機能やビデオ通話などの機能も含まれる．グループ一覧表示部はユーザが作成したグループの集合 $G \ni g_i (i = 0, 1, ..., n)$ の一覧および全てのユーザ U を示すグループ G_{all} が表示されている．ここで表示されているグループ一覧から一つ選びクリックで選択することで，グループ内メンバ一覧表示部に選択したグループ g_i のユーザ集合 U_i 一覧が表示される．グループ作成時はグループ名入力後に，グループ内メンバ一覧表示部に表示されているユーザ U_i から任意にユーザを選択し，グループを作成する．

ウィンドウ共有を行うためには，ツールバーからウィンドウ共有機能のボタンをクリックする．すると，管理画面に共有するウィンドウを選択するためのポップアップが表示されるので，そこから共有したいウィンドウを 1 枚以上選択し，決定する．また，ここで選択できるウィンドウは，管理画面を表示しているデスクトップ上に表示されているウィンドウとなる．グループ内のあるユーザのウィンドウを共有したい場合は，グループ内メンバ一覧に表示されているユーザをクリックする．すると同じように共有するウィンドウを選択するポップアップが表示されるので，同じように選択し，決定する．また，ウィンドウ共有はグループ内の全メンバ間で行われる．

4. 評価実験

本章では SFU 導入の効果を調べるために行なった評価実験について述べる．SFU 導入の目的は配信端末に高い負荷がかかることを防ぐことである．本研究ではこの負荷を配信に使用したメモリ量として計測し，SFU 導入以前のシステムと比較することで，SFU 導入によりどのような効果があったかについて調査した．

4.1 実験

本節ではメモリ使用量の計測方法と計測結果について述べる．まず，本システムを用いてウィンドウ共有を行う．その後，ウィンドウ共有が開始されてからの配信端末のメモリ使用量を計測し，SFU 導入前のシステムにおけるメモリ使用量と比較する．閲覧端末は 1 台から 15 台まで 1 台ずつ増加させ，各台数で新たにウィンドウ共有を開始してから 10s 間のメモリ使用量を計測した．ここで，10ms 間隔で計 1,000 回計測し，平均値および分散値を求めた．ただし，平均値についてはメモリ使用量の絶対値を，機能数やシステム内で管理するパラメータが大きく異なる旧システムと比較しても SFU 導入の効果を評価することは困難だと考え，各閲覧端末数における，閲覧端末の増加に伴うメモリ使用量の差を求めた．つまり，閲覧端末数が 1 台増加した際の，メモリ使用量の平均値の増減を調べた．本実験では共有するウィンドウのサイズは 960x1,200 とした．また，配信端末に用いた計算機のスペックについて表 4.1 に示す．

OS	macOS High Sierra
プロセッサ	3.4 GHz Intel Core i5
メモリ	16 GB 2400 MHz DDR4
グラフィックス	Radeon Pro 570 4096 MB

表 1: 実行環境

図 4 は計測したメモリ使用量の平均値の変化を示すグラフである．縦軸がメモリ使用量の平均値の差 [mbytes] を示し，横軸が閲覧端末数を示す．また，図 5 は計測したメモリ使用量の分散値 [mbytes$\hat{2}$] を示すグラフである．縦軸が分散値を示し，横軸が閲覧端末数を示す．また，各グラフにおいて，青色でバツ印のマーカーのグラフが SFU を導入した新システムのグラフであり，橙色で丸印のマーカーのグラフが SFU を導入する前の旧システムのグラフである．

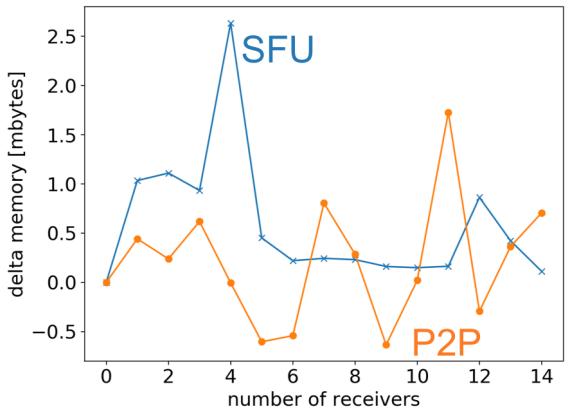

図 4: 平均値の変化

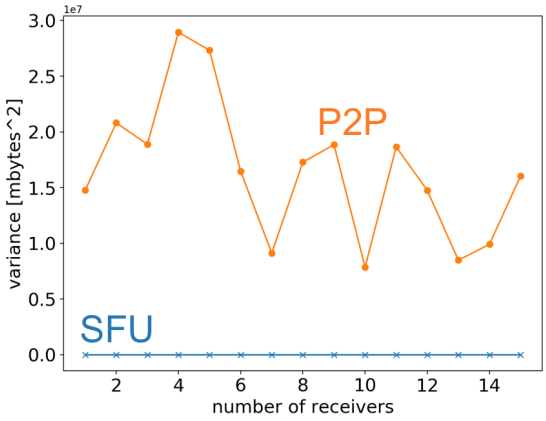

図 5: 分散値

4.2 考察

平均値の変化について，新システムでは台数が増えても，差の絶対値が常に 0 に近いことがわかる．一方で，旧システムではメモリ使用量の増減を繰り返しているが，閲覧台数が増えるに従いメモリ使用量の増加量が大きくなっている．本実験では閲覧端末を 15 台までとしたが，台数を増やすことでより顕著に違いが現れると考えられる．分散値については，旧システムは安定せず，一方で新システムでは常に安定し配信を行うことができるとわかった．

実験結果から，SFU の導入によって，負荷が高くなってしまう課題について解決することができるとわかった．特に 5 台から 11 台にかけてメモリ使用量の増加量はほぼ 0 であり，P2P での配信で発生する，閲覧端末数の増加に伴う配信端末の負荷の大きな増加を防ぐことができた．また，閲覧台数が 15 台を超え，その後も台数が増加すればするほど，SFU 導入の効果が現れると考えられる．一方で，閲覧台数が少ない場合には，メモリ使用量の大きな増加が見られた．これは，配信開始直後のエンコードやコネクション確立，その他配信に関する初期化処理などのプロセスにメモリが使用された結果であると考えられる．これに関して，メモリ使用量の計測開始のタイミングや，計測時間を見直す必要があると考えられる．また，SFU 導入の効果として，配信にかかる負荷の安定性の向上が挙げられる．このことから，突発的に PC が重くなり他のアプリケーションの利用が困難になるといったことがなくなり，使用性や効率性の向上が期待できる．

5. おわりに

本稿では複数ウィンドウ画面のグループ内共有システムとそのインタフェースについて述べた．本システムはアプリケーションウィンドウ画面共有を，ユーザ個人ではなくグループ単位で行うことで，共有を行う際に選択しなければいけない情報を削減し，円滑な共有開始手続きの完了を可能にした．また，WebRTC SFU を導入することで，閲覧端末の増加に伴う配信端末の使用メモリ量の増加を，P2P 通信を利用する場合と比較すると大きく抑えることができた．また，使用メモリ量も安定しており，本システムの使用性や効率性の向上が期待できる．

謝　辞

本研究の一部は JSPS 科研費 15K00422, 16K00420 の助成を受けたものです。

参考文献

[1] Nattha Buasri, Tanasak Janpan, Ularn Yamborisut and Damras Wongsawang. “Web-based interactive virtual classroom using HTML5-based technology”, Third ICT International Student Project Conference, pp.33-36, 2014.

[2] Huaying Xue and Yuan Zhang. “A WebRTC-based video conferencing system with screen sharing”, 2nd IEEE International Conference on Computer and Communications, pp.485-489, 2016.

[3] Hiroyuki Masaki, Hitoshi Habe, Nobukazu Iguchi. “Flexible Screen Sharing System between PC and Tablet for Collaborative Activities”, 10th International Conference on Complex, Intelligent, and Software Intensive Systems, pp569-571, 2016.

[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.

[5] Satoru Iwata, Tadachika Ozono and Toramatsu Shintani. “Any-Application Window Sharing Mechanism based on WebRTC”, 6th IIAI International Congress on Advanced Applied Informatics, vol.00, pp.808-813, 2017.