「マルチメディア，分散，協調とモバイル
(DICOMO2013)シンポジウム」 平成25年7月

リアルタイム/非リアルタイムウェブコミュニケーションをサポートするブラウザ同期方式

田坂和之[†1]　大岸智彦[†1]　井戸上彰[†1]

本論文では，ブラウザの表示画面やブラウザ上でのボタン押下などの操作を，複数のユーザ間でリアルタイム/非リアルタイムに共有する，ブラウザ同期方式を提案する．コールセンターや遠隔授業のようなサービスにて，従来方式では，円滑なコミュニケーションを目指し，オペレータがユーザのブラウザ操作を遠隔支援するブラウザ同期を実現する．さらに，提案方式は，オペレータの存在有無に関わらず，サービスを提供可能とする．具体的には，リアルタイムだけでなく，非リアルタイムなウェブコミュニケーションでもブラウザ同期を実現するため，オペレータによる操作内容や操作時刻を含む同期情報を保存し，コンテンツへ自動付与することによって，操作情報の共有時刻を変更する．このタイミングを，音声や映像のストリーミングコンテンツの出力時刻や通信環境(ネットワーク遅延やレンダリング性能)に応じて決定する．さらに，提案方式を評価するため，プロトタイプシステムを実装し，ブラウザ間での出力時刻の差や異種ウェブコンテンツの出力時刻の差を計測した．結果，その出力時刻の差は 300-400ms となり，提案方式は同期範囲の目安である時間(400ms)以内を実現できることがわかった．

A Browser Synchronization Method for Supporting Real and Non-Real-Time Web Communication

KAZUYUKI TASAKA[†1]　TOMOHIKO OGISHI[†1]　AKIRA IDOUE[†1]

1. はじめに

音声/映像会議サービスや遠隔操作サポートサービスなどのウェブコミュニケーションサービスが，インターネット上で広く受け入れられるようになってきた．ウェブコミュニケーションサービスは，コールセンターや遠隔授業のような多種多様なサービスで利用されている．これらのサービスにて，ユーザがブラウザ操作に不慣れであったとしても，オペレータによる遠隔からの代理操作のサポートにより円滑なコミュニケーションを実現することができる．

このようなウェブコミュニケーションについて，リアルタイムなウェブコミュニケーションと非リアルタイムなコミュニケーションに分類することができる．図 1 は，遠隔授業サービスの例を示している(Use Case 1：リアルタイムなウェブコミュニケーション，Use Case2：非リアルタイムなウェブコミュニケーション).

Use case 1: 先生は，自端末のブラウザを使って本日の授業用のウェブページを開く (図 1 (i)). 生徒 (Alice)は，自端末のブラウザを開き，受講の開始を要求すると，音声/映像メディアを含むストリーミングコンテンツの受信を開始すると同時に，先生が開いているウェブページと同じページへ誘導される (図 1 (ii)). Alice のブラウザは，先生のブラウザ上における操作内容(ボタン押下，描写，ページ遷移など)を受信し，同じ操作を実行する．(図 1 (iii)).

†1 (社)KDDI 研究所
KDDI R&D Laboratories Inc.

Use case 2: 生徒 (Bob)は，リアルタイムに受講することができなかったが，ブラウザを使って非リアルタイムな授業の受講を希望する．Bob のブラウザは，サーバに蓄積された音声や映像のストリーミングコンテンツの受信を開始すると同時に，先生が最初に開いたウェブページを取得し，出力する (図 1 (iv). Bob のブラウザは，先生がリアルタイムの授業においてブラウザを操作した場面の動画を表示する際，その操作を実行する(図 1 (v)).

図 1 の Use case 1 を実現するための方法が，これまで検討されてきた[1]-[3]. 従来方式[1]-[3]では，ブラウザ間でブラウザ操作の内容をリアルタイムに共有することによってブラウザの表示画面を同期している．さらに，ストリーミングコンテンツの出力時刻を同期する方式も検討されてきた[4]-[9]. 従来方式[4]-[9]は，円滑なコミュニケーションの向上を目的として，ストリーミングコンテンツなどの連続したデータパケットの出力時刻を同期可能である．従来方式は，ネットワーク遅延と各データの生成時刻から出力時刻を推定し，同種メディア(例.音声)や異種メディア(例.音声と映像)の出力時刻を変更して出力時刻の同期を実現する．

さらに，本論文では，リアルタイムだけでなく，Bob が非リアルタイムでの受講(あるいは Alice が受講した授業の復習)を対象としたサービス(Use case 2)を実現する，新たなブラウザ同期方式を提供する．本方式は，先生などのオペレータが離席している場合においても，遅延耐性のあるブラウザ操作共有を実現するとともに，オンデマンドで提供

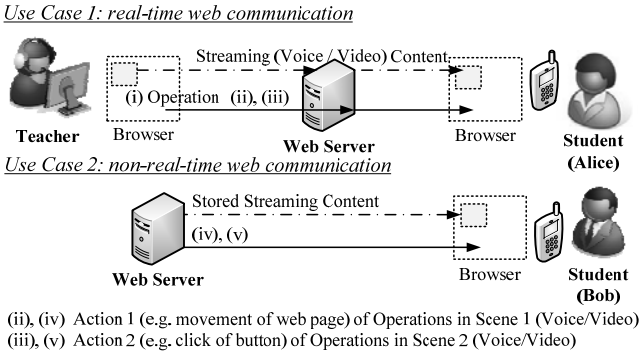

図 1.　利用シナリオ

するストリーミングの出力時刻にあわせてブラウザ操作を実行可能とする．本方式は，ストリーミングコンテンツとその他のウェブコンテンツ(非ストリーミングコンテンツ)の出力時刻を同期するため，ネットワーク遅延やレンダリング性能から各コンテンツの出力時刻を推定し，実際の出力時刻を変更する．これにより，ユーザは音声や動画で視聴している内容と，オペレータの操作に基づくウェブページの表示やマーカーの描画の一致を確認することができる．

以下 2 章では，関連する従来方式を述べるとともに利用シナリオを実現するための課題について述べる．3 章にてリアルタイム/非リアルタイムなウェブコミュニケーションでの円滑なコミュニケーションを実現するためのブラウザ同期方式を提案する．4 章では提案方式に基づいて実装したプロトタイプシステムの概要について述べ，各コンテンツの出力時刻の差などの観点から提案方式の性能を評価し，有効性を示す．最後に 5 章で本論文をまとめる．

2. 従来方式の概要と課題

2.1 従来のブラウザ同期方式

物理的に離れたユーザ間でブラウザの表示画面をリアルタイムに同期するブラウザ同期方式が，これまで検討されてきた[1][2]．従来方式[1]は，ブラウザ間でブラウザの操作内容を共有するだけでなく，同期のための操作権限の制御を可能とする．一方，従来方式[2]は，ユーザの移動を考慮している．ユーザが，移動にともないノートパソコンから携帯電話といったように使用する端末を変更すると，ウェブブラウザの情報(タブ，履歴，フォームなど)を引き継ぐことを可能とする．さらに筆者らも，リアルタイムなウェブコミュニケーションにおいて，通話相手のブラウザ間でブラウザの表示画面を同期するブラウザ同期方式を提案してきた[3]．従来方式[3]は，ネットワーク遅延やレンダリング性能が異なるブラウザ間でウェブコンテンツの出力時刻を同期することも可能とする．従来方式[3]は，他の従来方式と比較して以下の 3 つの利点を持つ．一つ目の利点は，通話相手と会話をしながら同じタイミングで同じウェブページや操作の実行結果を確認することができるため，円滑に会話することができる点である．二つ目の利点は，ユーザに対してブラウザへソフトウェアやプラグインのインストールを要求することがないため，ユーザはすぐにサービスを利用することが可能となる点である．三つ目の利点は，コールセンターや遠隔授業だけでなく，センサの遠隔監視など，多種多様なウェブサービスに活用できる点である．

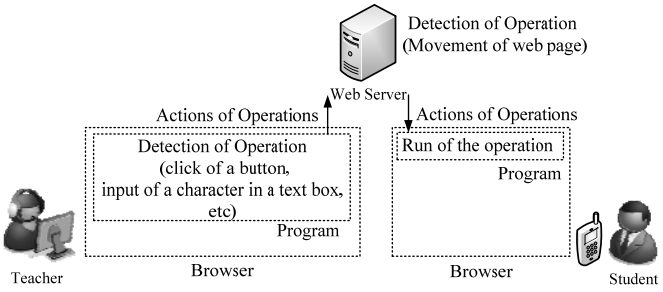

図 2.　従来方式を使った典型的なブラウザ同期モデル

図 2 は，ブラウザ同期の典型的なモデルを示している．ノートパソコンやスマートフォンなどの端末上の各ブラウザは，ウェブサーバからウェブコンテンツを取得し，表示する．コールセンターや先生などのオペレータがウェブページを操作すると，ウェブブラウザがその操作の種類(例．ウェブページの遷移，ボタンの押下，テキストボックスへの文字入力など)を検知する．ブラウザは，検知したブラウザの操作情報を他のブラウザと共有し，その出力を同期する．

上記のような非ストリーミングコンテンツとは別に，ストリーミングコンテンツの出力時刻を同期する方式が提案されてきた[4]-[9]．従来方式[4]-[9]は，あるストリーミングコンテンツにおいて，連続するデータパケットの生成時刻の間隔と出力時刻の間隔を一定に保つ．この時，各データのライフタイム内に出力するため，ビットレートの変更や帯域確保を行う．さらに，各方式は，映像と音声といったように異なるメディアでの出力時刻を同期する．このような同期は，ユーザの動作と話している内容をマッチするリップシンク技術として一般に知られている．

2.2 ブラウザ同期のための典型的なアーキテクチャ

図 3 は，本論文を通して想定するアーキテクチャを示す．図 3 のアーキテクチャは，従来のブラウザ同期方式をベースとしている．

本アーキテクチャは，ストリーミングコンテンツのためのブラウザ同期モジュール(図 3 (a))を備える．本モジュールは，ネットワーク遅延に応じて音声/映像コンテンツの出力時刻を同期する．一方，非ストリーミングコンテンツのためのブラウザ同期モジュール(図 3 (b))は，ウェブコンテンツや操作情報を送信するタイミングやそれらを出力する

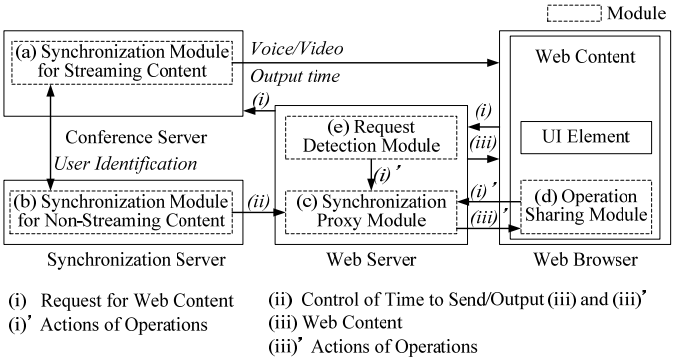

図 3.　ブラウザ同期用のアーキテクチャ

時刻を決定する．本モジュールは，ウェブサーバ上のブラウザ同期用のプロキシーモジュール(図 3 (c))と共有する. 本モジュールは，ブラウザ上の操作共有モジュール(図 3 (d))によって共有されるべきブラウザの操作情報を収集する．操作共有モジュール(図 3 (d))は，ブラウザ同期用のプログラムとして JavaScript のようなスクリプトを読み込む. また，ウェブページの遷移要求検知モジュール(図 3 (e))は，ブラウザからのウェブページの遷移要求を監視・検知する．

2.3 想定環境下での従来方式の課題

図 3 に示したアーキテクチャ内で実行される従来方式の課題を以下に示す.

従来方式は，先生と生徒といったようにユーザ間で現在開いているウェブページや操作情報をリアルタイムに共有する．この時，各ブラウザは，同期用の情報を共有するためのウェブセッションをウェブサーバと事前に確立する必要がある．したがって，オペレータが離席し，ブラウザのセッションが確立されていない状況下では，生徒のようなユーザは，ブラウザの操作情報をオペレータと共有することが困難である (課題 1).

ウェブサーバが操作情報の通知タイミングを変更することによって，ユーザのブラウザはオペレータ不在の場合にもブラウザ情報を共有することが可能となる．ストリーミングコンテンツに関しては，ユーザの要求に応じてウェブサーバが事前に音声/動画コンテンツを録音/録画し，ユーザへ提供することが可能となる．しかしながら，従来方式は，ブラウザの操作情報を自動的に記録し，ブラウザ上で逐次的に実行することが困難である(課題 2).

ブラウザが異なるウェブコンテンツ(例.ストリーミングコンテンツや非ストリーミングコンテンツ)を受信・出力するとき，ブラウザ内やユーザ間で，各コンテンツの到達遅延や出力遅延が異なっている．このような環境下で，同期はずれが発生しやすい(課題 3). 同期はずれでは，ユーザに対して，オペレータの声や動作内容と，非ストリーミングコンテンツの表示内容が異なるため，コミュニケーションにおいて違和感を与える原因となる.

3. ブラウザ同期方式の提案

本論文では，図 3 に示したアーキテクチャを基にリアルタイムなウェブコミュニケーションだけでなく，非リアルタイムなウェブコミュニケーションでのブラウザ同期を実現する方式を提案する．本方式は，コールセンターや遠隔授業のような多種多様なウェブコミュニケーションサービスでのユーザ満足度の向上に必要な技術となる.

3.1 課題を解決するための要件

2.3 節で示した 3 つの課題を解決するためのブラウザ同期方式を実現するための機能要件と性能要件を以下に示す.

機能要件 1:オペレータが不在時にも操作情報をユーザへ通知可能とするため，ブラウザ上の操作内容と操作時刻を自動的に記録する必要がある．さらに，記録した操作内容と操作時刻を基に，ユーザのブラウザにて逐次的に実行する機能が必要である.

機能要件 2：オペレータの対応可否に応じてブラウザの操作情報を共有する方法を変更するため，ウェブコンテンツへ埋め込むスクリプトを自動的に変更可能とする機能が必要である.

機能要件 3：オペレータが不在の場合においても同期はずれを回避し，異類ウェブコンテンツ間の出力時刻を同期する必要がある.

上記の機能要件に加えて，ユーザに同期はずれを実感させない性能要件を以下に示す.

性能要件 3：，[11][12]の結果から目標時間となる 400ms 以内で異種ウェブコンテンツ間の出力時刻の差を 400ms とする必要がある. もし，出力時刻の差が 400ms を超えると，オペレータの音声/映像の内容と，オペレータによるブラウザ上での操作内容が一致しない状況が発生してしまうため，ユーザが授業内容を理解できなくなる恐れが生じる.

3.2 典型的なアーキテクチャで実現する機能

図 3 のアーキテクチャ上で 3.1 節に示した要件を満たす 3 つの機能を以下に示す.

機能 1: 遅延耐性ブラウザ共有機能

機能要件 1 を満たすため，ウェブサーバがオペレータのブラウザ操作を監視し，検知した操作情報をデータセットとして記録する．ウェブサーバは，ユーザのブラウザにてブラウザの操作情報を逐次的に実行するため，記録したデータセットを基に各ブラウザ情報の通知タイミングを変更する.

機能 2：同期方法変更機能

機能要件 2 を満たすため，ウェブサーバやブラウザが，オペレータの対応可否の変化に応じて，従来のブラウザ同期を実現するためのスクリプトをウェブコンテンツに埋め込むか，ブラウザの操作情報を集約したスクリプトをウェブコンテンツに埋め込むかを判定する．さらに，判定結果に応じて同期方法を決定し，ウェブコンテンツへスクリプトを自動で埋め込む．

機能 3：異種ウェブコンテンツ間のブラウザ同期機能

機能要件 3 と性能要件 1 を満たすため，本機能は，リアルタイムウェブコミュニケーション時にブラウザを操作した時刻に出力したストリーミングデータの出力時刻を記録する．さらに，非リアルタイムウェブコミュニケーションにおいて，ストリーミングデータの出力時刻とブラウザ操作の出力時刻を一致させる．

3.3 遅延耐性ブラウザ共有機能

本機能は，非リアルタイムウェブコミュニケーションにおいてブラウザの操作情報の共有を遅らせることにより同期を実現可能とする．具体的には，本機能が操作情報を検知すると，その情報をウェブサーバにて一時的に記録し，非リアルタイムなウェブコミュニケーションを要求するユーザに対して記録した操作情報を通知する．図 4 と図 5 は，それぞれリアルタイムウェブコミュニケーションと非リアルタイムウェブコミュニケーションのシーケンスを示している．

図 4 では，ウェブサーバと各ブラウザが従来方式と同様な処理を実行する．各ブラウザは，ウェブセッションの ID (UID)を使ってウェブページを取得する(図 4 (1)-(2))．ウェブサーバはブラウザの操作情報を共有するユーザをグループ化するための識別子(ルーム ID)を取得すると，ウェブサーバは UID とルーム ID を紐付ける(図 4 (3)-(4))．その後，ブラウザは，そのルーム ID に複数のユーザが存在することを確認すると，ブラウザ同期を開始できることを検知する(図 4 (5))．オペレータは，ブラウザ同期の開始要求のための HTTP メッセージをウェブサーバへ送信し，ブラウザ同期を開始する(図 4 (6))．オペレータがウェブページを操作すると，ブラウザやウェブサーバが操作内容を検知し，他のユーザと共有する(図 4(7)-(13))．

提案方式では，ウェブサーバは，ブラウザや自サーバが操作情報を検知すると，操作内容と操作時刻を記録する(図 4 (8')，図 4 (12'))．ブラウザは，オペレータの指示によりブラウザ同期を終了する HTTP メッセージをウェブサーバへ送信することによって，ブラウザ同期を終了する(図 4 (14)-(15))．この時，ウェブサーバは，同期を開始してから

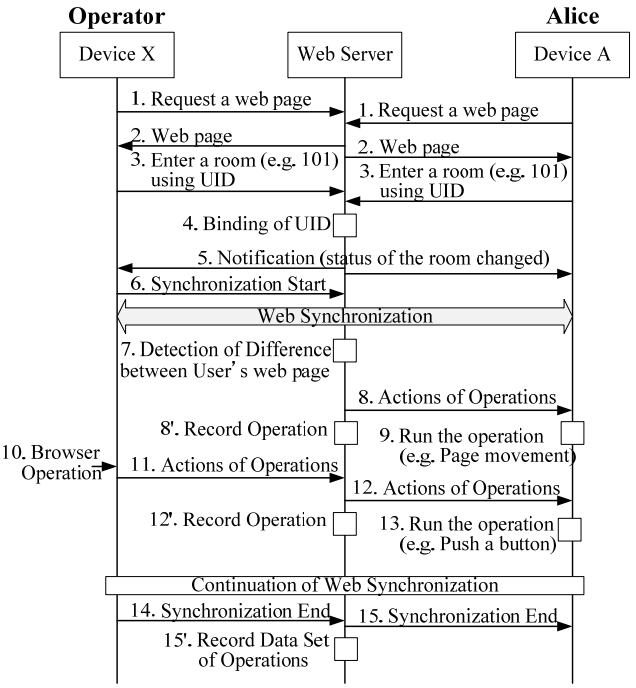

図 4.　リアルタイムウェブコミュニケーションでのブラウザ同期シーケンス

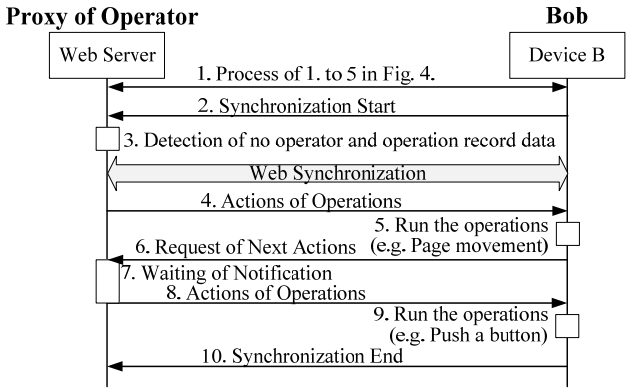

図 5.　非リアルタイムウェブコミュニケーションでのブラウザ同期シーケンス

終了するまでに記録した操作内容と操作時刻を一つのデータセットとして，識別子とともに記録する (図 4 (15'))．

非リアルタイムウェブコミュニケーションでは，ウェブサーバは，図 4 でのオペレータの代わりにユーザのブラウザを遠隔操作する．ウェブサーバが UID とルーム ID を紐付けるまでの最初のプロセスは図 4 と同じである (図 5 (1))．ユーザのブラウザは，ブラウザ同期の開始を要求する(図 5 (2))．ウェブサーバは，オペレータの対応不可を検知し，操作情報のデータセットを取得する(図 5 (3))．ウェブサーバは，ブラウザの操作内容を逐次的に実行するためのスクリプトをウェブコンテンツへ追加し，ブラウザへ返信する(図 5 (4))．スクリプトの埋め込み方法については，同期方法変

更機能にて後述する．ブラウザは，受信したブラウザの操作を実行する(図 5 (5))．操作実行後，ブラウザは次の操作内容をウェブサーバに要求する(図 5 (6))．ウェブサーバは，リアルタイムウェブコミュニケーションと同じ間隔で操作内容を実行するようブラウザに要求する(図 5 (7)-(8))．ブラウザは，全ての操作を実行すると，ブラウザ同期を停止する(図 5 (9)-(10))．

ブラウザ同期実行中(図 5 (4)-(10))，ユーザが他のウェブページの遷移や操作を実行すると，本機能は記録した操作を継続実行するため，変更される前の状態に戻す．

3.4 同期方法変更

機能要件 2 を満たすため，本機能は，オペレータの対応可否に応じて，ウェブコンテンツへ埋め込むブラウザ同期用のスクリプトを自動的に変更する．

ウェブコンテンツの提供者は，ブラウザの操作内容や操作時刻を検知，ウェブサーバ経由でブラウザと共有するスクリプト(例. JavaScript)をウェブコンテンツへ埋め込む．本スクリプトにより，ユーザは追加のソフトウェアやプラグインのインストールを行わなくて済む．

ウェブコンテンツへの JavaScript の埋め込み箇所を図 6 に示す．ウェブコンテンツの提供者は，ウェブコンテンツの HTML Body 部(図 6 (i))内にボタンやテキストエリアなどのユーザインタフェース(UI)要素のコードを作成する．次に，UI 要素(図 6 (iii))を同期用 JavaScript(図 6 (ii))の画面入力イベント処理部内に追加する．ウェブブラウザは，ウェブコンテンツ読み込み時に UI 要素の一覧をウェブサーバ内の同期制御モジュールに登録する．登録後，ユーザが UI 要素を操作すると，同期用 JavaScript API 間で操作内容を送受信し，操作内容を同期する．スクロールの位置同期やボタンの色を変更するといった操作内容のように，UI 要素のタイプを追加したい場合，ウェブコンテンツの提供者は，新たに UI 要素の ID や操作を定義可能である．例えば，スクロール位置の同期であれば，以下のように定義する．

(document.getElementById (“scroll”), “scroll”, “10”)

さらに，本機能は，非リアルタイムウェブコミュニケーションの場合，操作時刻を基に各操作を実行するスクリプトを並べる．ブラウザがこのウェブコンテンツを読み込むと，ウェブサーバの問い合わせ後，逐次的に実行し，リアルタイムウェブコミュニケーションで実行していた操作を再現する．

3.5 異種ウェブコンテンツ間のブラウザ同期機能

本機能は，ストリーミングコンテンツと非ストリーミングコンテンツといった異種ウェブコンテンツの出力時刻を同期する．例えば，図 1 での Use case 2 での利用を想定している．図 7 は，ストリーミングコンテンツと非ストリーミングコンテンツのブラウザ同期のシーケンスを示している．

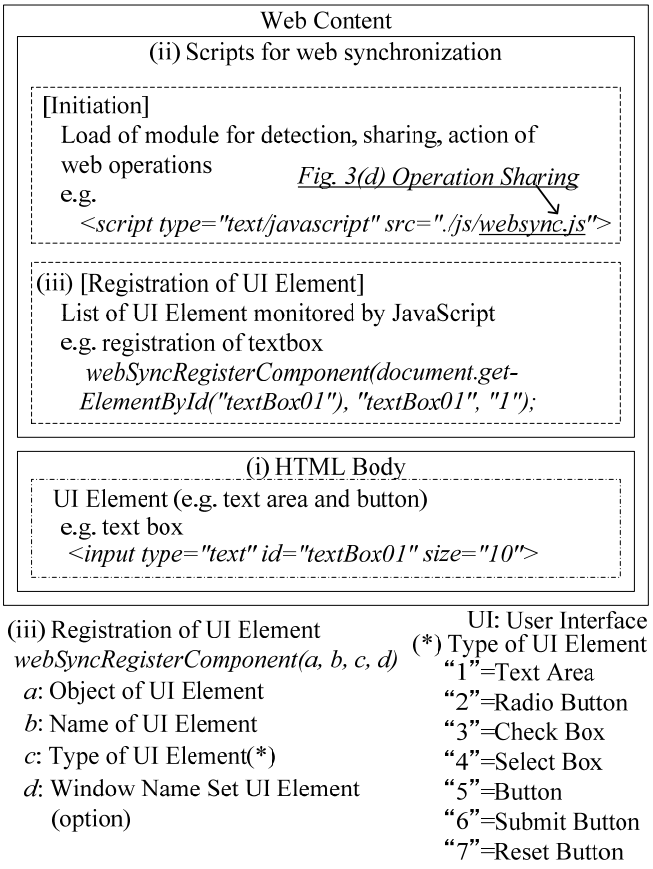

図 6.　ブラウザ同期用のスクリプトの埋め込み方法

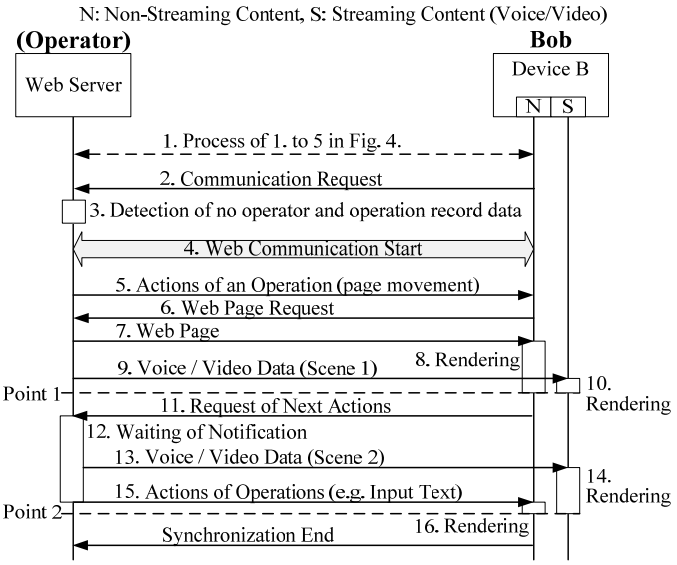

図 7.　ストリーミング/非ストリーミングコンテンツのための同期シーケンス

図 7 のシーケンスでは，ウェブサーバと端末(ブラウザ)は，最初に図 5 (1)-(3)で述べた処理を実行する(図 7 (1)-(3))．ウェブサーバは，ブラウザに対して，ブラウザ同期を開始

するとともに，ストリーミングの配信を開始する．その後，ウェブサーバは，ブラウザの操作情報の送信を開始する(図 7 (4))．

本機能は，ストリーミングコンテンツ内のオペレータの会話の内容や場面(図 7 Scene 1，Scene 2)とブラウザの操作が合うように，ストリーミングの出力時刻に合わせて，該当するブラウザの操作内容を実行する(図 7 Point 1, Point 2)．このため，ウェブサーバは，ブラウザの操作情報を送信するタイミングを変更する．

図 7 での Point 1 では，ウェブサーバは，音声/映像コンテンツの Scene 1 のデータよりも前にブラウザの操作情報(例．ページ遷移)を送信する (Fig. 7(5))．ブラウザは，ウェブサーバからの指示により新しいウェブページへ遷移する(図 7 (6)-(7))．ブラウザは，Point 1 で Scene 1 の音声/映像コンテンツの出力時刻とページ遷移の表示時刻を一致させる(図 7 (8)-(11))．ここで，異種ウェブコンテンツを同時に送信すると，Scene 1 の音声/映像コンテンツが表示され，別の場面に移った後，ページ遷移を実行することになる．

図 7 での Point 2 では，ウェブサーバは，音声/映像コンテンツの Scene 2 のデータよりも後にブラウザの操作情報(例．文字入力)を送信する (図 7 (12)-(15)). ブラウザは, Point 2 で Scene 2 の音声/映像コンテンツの出力時刻とページ遷移の表示時刻を一致させる(図 7 (13)-(14)，(16))．ここで，異種ウェブコンテンツを同時に送信すると，Scene 2 の音声/映像コンテンツが表示されるより前の場面で，場面とは関係しないテキストを表示することになる．

ウェブサーバからブラウザへの操作情報の通知タイミングを決定するための 3 つのプロセスを以下に示す．

(Process 1) 初期パラメータ値の取得(図 8 (1)-(3))

最初のプロセスにて，ウェブサーバは，ネットワーク遅延(ウェブサーバがウェブコンテンツを送信してからブラウザでそのウェブコンテンツを受信するまでの時間)ならびにウェブコンテンツのレンダリング時間(ウェブコンテンツを受信してから表示するまでの時間)をブラウザから収集する．ウェブサーバは，アップロードスループット，ダウンロードスループット，レンダリング性能をネットワーク遅延やレンダリング時間から推定する．詳細を以下に示す．

ユーザがウェブサーバへアクセスすると，ウェブサーバや同期サーバは, 初期パラメータの取得を開始する(図 8 (1))．ブラウザは，最初に 3 つのパラメータ値をウェブサーバへ送信する(図 8 (2))．3 つのパラメータとは，ウェブコンテンツを要求するメッセージのアップロードに要する遅延(アップロード遅延)($T_{ud\_i-1}^{(D)}$)，ウェブコンテンツのダウンロード遅延($T_{dd\_i-1}^{(D)}$)とレンダリング時間($T_{rt\_i-1}^{(D)}$)である．なお，D と i は，それぞれ，ウェブコンテンツ名の識別子とシーケンス番号を示している．

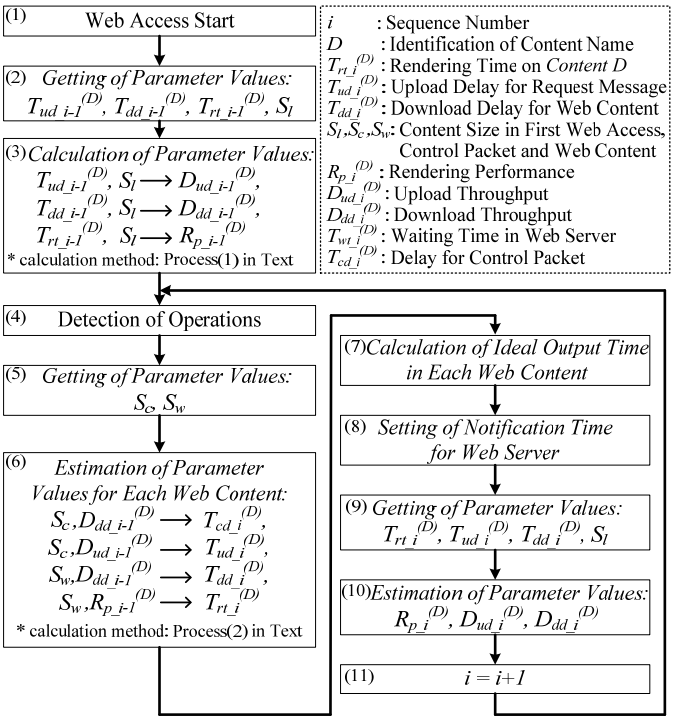

図 8. ウェブコンテンツの出力時刻を決定するフローチャート

同期サーバは，アップロード遅延，ダウンロード遅延ならびにウェブコンテンツのサイズ(S_l)から，アップロードにおけるスループット($D_{ud\_i-1}^{(D)}$ ($=S_l / T_{ud\_i-1}^{(D)}$))とダウンロードにおけるスループット($D_{dd\_i-1}^{(D)}$ ($=S_l / T_{dd\_i-1}^{(D)}$))を計算する(図 8 (3))．さらに，同期サーバは，レンダリング時間とウェブコンテンツのサイズから，レンダリング性能($R_{p\_i-1}^{(D)}$ ($=T_{rt\_i-1}^{(D)} / S_l$))を測定する．

(Process 2) 各パラメータ値の推定(図 8 (4)-(6))

2 番目のプロセスにおいて，ウェブサーバは，プロセス 1 で取得した各パラメータから，ダウンロード遅延，アップロード遅延やレンダリング性能を推定する．同期サーバは，記録したデータセットを基にウェブサーバから操作情報を検知する(図 8 (4))．同期サーバは，ウェブサーバから，これから送信する制御パケットやウェブコンテンツのサイズ(S_c, S_w)を取得する(図 8 (5))．同期サーバは，操作情報を共有するための制御パケットのダウンロード遅延($T_{cd\_i}^{(D)}$ ($=S_c / D_{dd\_i-1}^{(D)}$))やウェブコンテンツを要求するための制御パケットのアップロード遅延($T_{ud\_i}^{(D)}$ ($=S_c / D_{ud\_i-1}^{(D)}$))を推定する(図 8 (6))．さらに，同期サーバは，ウェブコンテンツのダウンロード遅延($T_{dd\_i}^{(D)}$ ($=S_w / D_{dd\_i-1}^{(D)}$))やレンダリング性能($T_{rt\_i}^{(D)}$ ($=R_{p\_i-1}^{(D)} \times S_w$))を推定する(図 8 (6))．

(Process 3) 理想出力時刻と通知時刻の決定(図 8 (7)-(11))

3 番目のプロセスでは，ウェブサーバは，ウェブコンテンツの理想的な出力時刻を計算する．理想的な出力時刻とは，各コンテンツの出力時刻の差が，同期はずれを回避するための目標時間内になる出力時刻を示す．ウェブサーバは，ブラウザが新しいウェブページへ遷移したり，ブラウザの操作を理想的な出力時刻で表示する時刻から逆算して操作情報を通知する時刻を計算する．詳細を以下に示す．

非ストリーミングコンテンツの理想的な時刻は，ストリーミングの実際の出力時刻(T_{np1_1} ($=T_{ns1\_1}$ + (T_{rp1_1} - T_{rs_1})))となる(図 8 (7), 図 9 Point 1). 同期サーバは，非ストリーミングコンテンツを理想的な出力時刻が実際の出力時刻に一致するように，操作情報の送信タイミング($=T_{np1\_1}$ - ($T_{cd\_i}^{(D)}$ + $T_{ud\_i}^{(D)}$ + $T_{dd\_i}^{(D)}$ + $T_{rt\_i}^{(D)}$))を決定する．

本機能は，各パラメータを随時更新することによって，出力時刻の同期を継続する．ウェブサーバは，ウェブコンテンツのレンダリング後，ネットワーク遅延やレンダリング時間を取得し，スループットやレンダリング性能を更新する(図 8 (9)-(11))．もし，ブラウザがウェブコンテンツを取得中，ネットワーク遅延が変化した場合，ウェブサーバとブラウザ間の理想的な時刻の共有により，時刻同期の範囲[10]-[11]内でレンダリングを開始する時刻を変更する．

4. プロトタイプシステムの概要と性能評価

4.1 プロトタイプシステムの概要

提案方式を備えるプロトタイプシステムを実装した．実装環境を以下に示す．ウェブサーバおよび通話サーバを Redhat エンタープライズ 5 64bit 版上に実装した. 各サーバのハードウェアとして，CPU は Xeon 3.0 x4 を，メモリを 4GB 使用した．

また，通話サーバにおける SIP ソフトウェアとして Asterisk [12]を使用した．ウェブブラウザの操作内容を端末からウェブサーバ(同期制御モジュール)へ送信するためのツールとして Ajax (Asynchronous JavaScript and XML)を，ウェブコンテンツの取得をウェブサーバ(同期制御モジュール)から端末へ要求するためのツールとして Comet (Reverse Ajax)を使用した．音声/映像メディアを含むストリーミングコンテンツの送受信を Real-time Transport Protocol (RTP) [13]で開始するためのプロトコルとして，Session Initiation Protocol (SIP) [14]を使用した．

プロトタイプシステムを使用して，提案方式の性能を評価するため，図 10 に示す実験用のテストベッドを構築した．オペレータと生徒(Alice)がウェブコミュニケーションを開始するとき，それぞれ Device X と Device A を使用する．一方，生徒(Bob)がウェブサーバとウェブコミュニケーションを開始するとき，Device B を使用する．

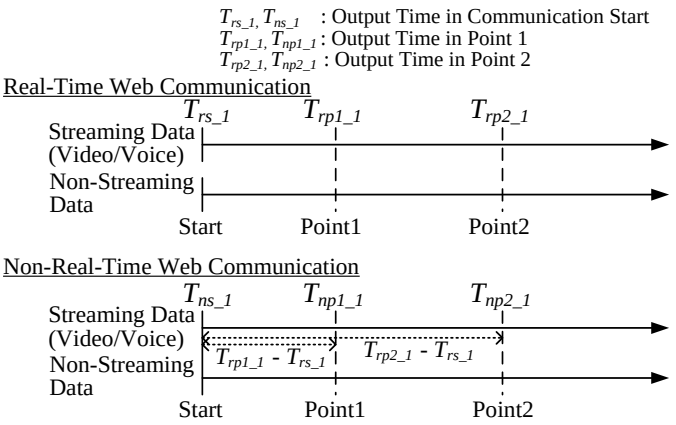

図 9. ストリーミングコンテンツのデータと非ストリーミングコンテンツのデータの出力時刻の関係

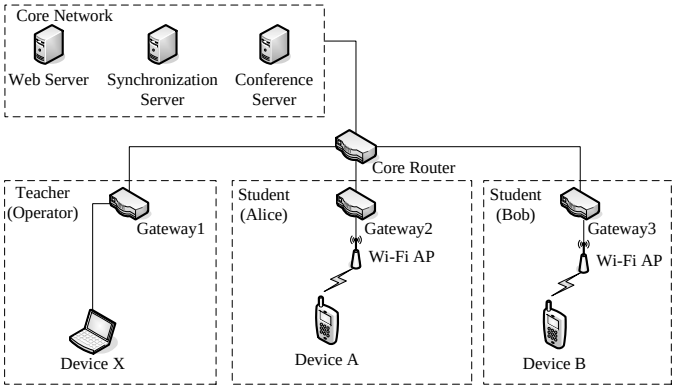

図 10. 実験用のテストベッド

4.2 性能評価のための測定項目

提案方式がリアルタイムなウェブコミュニケーションや非リアルタイムなウェブコミュニケーションの各シーケンス(それぞれ図 4，図 5 と図 7)にて性能評価 1 を満たすことができることを確認するための測定項目を以下に示す．

1) リアルタイム/非リアルタイムウェブコミュニケーションでの同期性能

本測定では，提案方式と 2 章で述べた従来方式を使用し，先生と Alice のブラウザ間でウェブコンテンツの出力時刻の差を測定する．さらに，非リアルタイムウェブコミュニケーションにおいて，リアルタイムウェブコミュニケーションの再現性を確認するため，Bob のブラウザで実行する操作内容の表示時刻の間隔と Alice が持つブラウザでの操作内容の表示時刻の間隔における差を測定する．

測定項目 1 : 本測定項目において，図 10 における Device A あるいは Device B のアクセスネットワーク(WLAN)の帯域幅を，Gateway 2 や Gateway 3 にて，1 Mbps から 20Mbps (Wi-Fi アクセスポイントでの実行スループットの限界値)に変更した．ウェブコンテンツのサイズは，3Mbyte に設定して測定した．

測定項目 2：本測定項目において，ウェブサーバ上のコンテンツサイズを 200KByte から 3MByte へ変更するとともに，スループットを固定した(Device X: 100Mbps, Device A and Device B: 1Mbps).

これらの測定結果によって，データ到達遅延やレンダリング時間が変化した場合においてもブラウザ同期を実現できることを評価する.

2) ストリーミングコンテンツと非ストリーミングコンテンツの出力時刻の同期性能

測定項目 3：Device B は，ストリーミングコンテンツと非ストリーミングコンテンツの受信を開始する．本測定項目では，ストリーミングコンテンツのデータの出力時刻と非ストリーミングコンテンツのデータの出力時刻を，機能 1 と機能 2 のみを実装した提案方式(1)と全ての機能を実装した提案方式(2)を使用して測定した.

本測定結果に基づき，異種ウェブコンテンツがブラウザ同期に与える影響および機能 3 の性能を評価する.

4.3 測定結果と性能評価

1) リアルタイム/非リアルタイムウェブコミュニケーションでの同期性能

図 11 と図 12 は，それぞれ測定項目 1 と測定項目 2 の測定結果を示している.

測定結果 1：図 11 は，ブラウザ間のスループットの差に応じてウェブコンテンツの出力時刻の差を示している．提案方式と従来の同期機能を持つ従来方式(2)の測定結果である出力時刻の差は，帯域幅(スループット)の差に関わらず，約 300ms となった．一方，同期機能なしの従来方式(1)の測定結果は，400ms を越えていた.

測定結果 2：図 12 は，レンダリング時間を増加させた場合における，ウェブコンテンツの出力時刻の差を示している.

提案方式と従来の同期機能を持つ従来方式(2)の測定結果である出力時刻の差は，コンテンツサイズ(レンダリング時間)の変化に関わらず，約 300ms となった．一方，同期機能なしの従来方式(1)の測定結果である出力時刻の差は，400ms を越える結果となった.

測定結果 1 および測定結果 2 は，提案方式がリアルタイムウェブコミュニケーションでは従来方式(2)と同等の性能を持つだけでなく，非リアルタイムなウェブコミュニケーションでもブラウザ同時を実現できていることを示している.

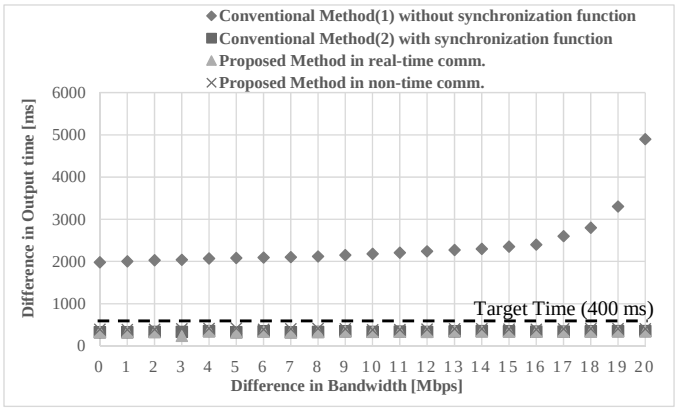

図 11. デバイス間における帯域幅の差の増加に応じたブラウザ間でのウェブコンテンツの出力時刻の差

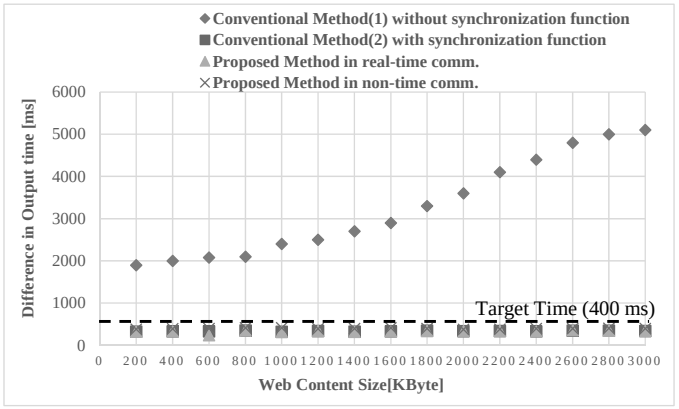

図 12. ウェブコンテンツイズの差の増加に応じたブラウザ間でのウェブコンテンツの出力時刻の差

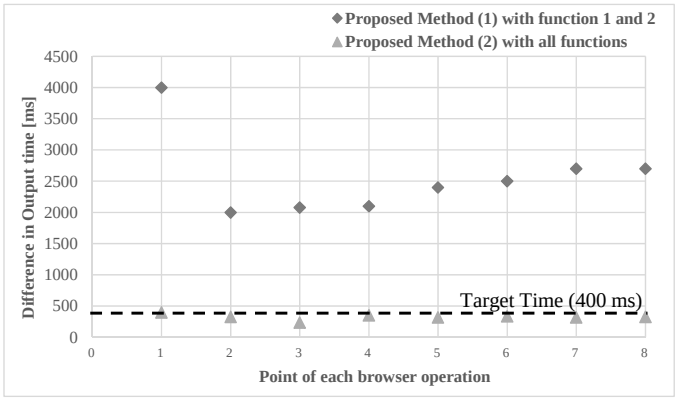

図 13. ストリーミングコンテンツのデータと非ストリーミングコンテンツのデータでの出力時刻の差

2) ストリーミングコンテンツと非ストリーミングコンテンツの出力時刻の同期性能

測定結果 3：図 13 は，先生が 8 つのブラウザ操作(Point 1 から Point 8)を実行した場合において，各操作における異種ウェブコンテンツの出力時刻の差の結果を示している．各操作は以下のとおり：1. ページ遷移，2. テキストエリア

への文字入力，3. ラジオボタンの選択，4. チェックボックスの選択, 5. セレクトボックスの選択, 6. ボタン押下, 7. サブミットボタンの押下，8. リセットボタンの押下.

全ての提案機能を持つ提案方式(2)の測定結果は，約400ms であった．一方，機能 1 と機能 2 のみを持つ提案方式(2)の測定結果は，400ms を超える結果となった．特に，Point1 のページ遷移の結果では，出力時刻の差が他の操作と比較して増加していた．これは，ブラウザが新しいウェブページを取得するためのステップ数が他の操作のシーケンスのステップ数と比較して増加するためである.

上記の結果から，提案方式は，3.1 節で述べた性能要件を満たすことができるといえる.

測定結果 1〜3 の結果から，提案方式は全ての要件を満たすことを示した．さらに提案方式は，グループ通信にも適用可能であるが，ブラウザ同期の対象となるデバイスの数を増加させた際の規模性や可用性などの評価を行う必要がある.

5. おわりに

本論文では，リアルタイムウェブコミュニケーションだけでなく，非リアルタイムウェブコミュニケーションにおけるブラウザ同期方式を提案した．提案方式では，ネットワーク遅延やレンダリング性能に応じて異種ウェブコンテンツ(ストリーミングコンテンツと非ストリーミングコンテンツ)の出力時刻を同期することを可能とする. 提案方式は, 従来のブラウザ同期方式と比較して以下の利点を持つ. 一つ目は，ユーザはオペレータが不在の際にも遠隔操作の支援サービスを利用することができる点である. 二つ目は，ウェブコンテンツの種類に関わらず，異種コンテンツを違和感なく視聴できる点である．さらに，プロトタイプシステムを実装し，提案方式の性能を評価した．具体的には，提案方式と従来方式においてウェブコンテンツの出力時刻の差を測定した．測定した結果は，提案方式はリアルタイムや非リアルタイムに関わらず, 300〜400ms という結果になり，目標時間(400ms)以内になることを示した.

最後に, 日頃ご指導頂く(株)KDDI 研究所中島所長ならびに阿野部門長に感謝する.

参考文献

1) L. Hwan-Gu, K. Won-Tae, K. Sun-Ja and L. Cheol-Hoon, “Design and Implementation of Mobile Cobrowsing Service which supports the sharing of web page among mobile users,” Proc. IEEE conf. Convergence and Hybrid Informatin Technology (ICHIT’08), pp.266-269, 2008.

2) R. Valle, A. Passito, R. Novellino and A. Penaranda, “Synchronization Web Browsing Data with Browserver,” Proc. IEEE Symp.Computers and Communcations (ISCC2010), pp.738-743, 2010.

3) K. Tasaka, T. Ozu and A. Idoue, “A Novel Web Synchronization Method for Supporting Smooth Web Communication, Proc. IEEE conf. Advanced Information Networking and Applications (AINA2013) Workshop (FINA2013), Mar. 2013.

4) Y. Ishibashi and S. Tasaka, “A Media Synchronization Mechanism for Live Media and Its Measured Performance,” IEICE transaction in Japan, E81-B(10), 1998.

5) O. Wongwirat and S. Ohara, “Haptic media synchronization for remote surgery through simulation,” IEEE Trans. Multimedia, vol.13, no.3, pp.62-69, 2006.

6) P. Wagner and P. Frossard, “Playback delay and buffering optimization in scalable video broadcasting,” Proc. IEEE conf. Multimedia Services Access Network (MSAN’05), pp.746-752, 2005.

7) Y. Ishibashi and S. Tasaka, “A group synchronization mechanism for live media in multicast communications,” Proc. IEEE conf. Global Communications (Globecom’97), pp.746-752.

8) Y. Ishibashi and S. Tasaka, “A distributed control scheme for causality and media synchronization in network multimedia games,” Proc. IEEE conf. Computing Communication and Network (ICCCN2002), pp.144-149, 2002.

9) K. Tasaka, N. Imai, M. Isomura and A. Idoue, “A media synchronization method for real-time group communication in a multiple device environment,” Proc. IEEE conf. Intelligence in Next Generation Networks (IMSA2008), pp.1-6, 2008.

10) ITU-T Recommendation G.114, “One-way transmission time”, 2003.

11) R. Steinmz, “Human perception of jitter and media synchronization”, IEEE Journal on Selected Areas in Communication, 4(1), 1996, 61-72.

12) Digium, “Asterisk: The open source PBX and telephony platform,” 2006.

13) H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “A Transport Protocol for Real-time Applications,” RFC1889, 1996.

14) J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and S. Schooler, “SIP: Session Initiation Protocol,” RFC3261, 2002.