Electronic Preprint for Journal of Information Processing Vol.22 No.3

Regular Paper

A Web Synchronization Method for Browser-Based Communications

Kazuyuki Tasaka[1,a)] Tomohiko Ogishi[1]

Received: September 16, 2013, Accepted: February 14, 2014

Abstract: In this paper, we propose a web synchronization method (WSM) to share operation data on a browser and synchronize output time of data among browsers in browser-based communications such as video conferencing and remote control services. WSM continually provides users with an environment for smooth browser-based communications even if users are in a heterogeneous environment where network delay and rendering time among browsers fluctuate. This fluctuation causes the difference of output time among browsers and a lack of synchronization (out-of-synchronization). This is perceived as being somewhat strange, or even annoying. Several methods have been studied to prevent out-of-synchronization for streaming content such as video and voice data. WSM synchronizes the output time of streaming content and/or non-streaming content after sharing browser operations (e.g., page movement) among conversational partners. WSM also maintains synchronization of the output time even if a device is connected to different access networks during a conversation. Synchronized output is realized by controlling the time to notify each browser of browser operations, and by controlling the time to send and output web content according to the network delay and rendering performance. For considering feasibility, WSM works on a web browser and does not need additional software. We implemented a prototype system and measured the difference in the output time among browsers. The results show that WSM achieves web synchronization within 300 ms while the target time was 320 ms.

Keywords: web synchronization, browser-based communications, streaming and non-streaming communications

1. Introduction

Communication services using both voice and visual (e.g., video and web) media are growing not only in fixed network environments but also in mobile network environments. In communication services, voice media plays a fundamental role enabling multiple users to talk with each other in real-time. Visual media compensates for difficulties in communication and contributes to a smooth communication, for example, in the case that a user wishes to instruct the operation of an item to others.

Figure 1 depicts a use case scenario where a smooth communication is necessary. At first, two users, Bob and Carol are consulting a travel agency (Alice) for their travel plans using a PC and a mobile phone, respectively as shown in Scene 1. Then, Alice as the telephone operator of the travel agency recommends plans by sharing web contents in addition to giving an oral explanation as shown in Scene 2. The timings of showing the web contents and operating actions such as highlighting a button should be synchronized by devices of the operator and users. In the case of the mobile phone, the synchronization should be continually available even if the connected network environment is changed, for example from Wi-Fi to 3G, during the communication as shown in Scene 3.

The existing techniques [1], [2], [3] realize the sharing of web contents. The users run a browser and share the operations (e.g., webpage movement) among browsers using additional software or a browser plugin. However, these techniques do not consider the sharing among conversational partners of voice/video communications. Moreover, these techniques do not focus on the synchronization of the output time of a webpages. Therefore, it gives users a feeling of wrongness due to the difference of output times among browsers in a heterogeneous environment where the network delay and the rendering time among browsers fluctuate. This difference causes out-of-synchronization.

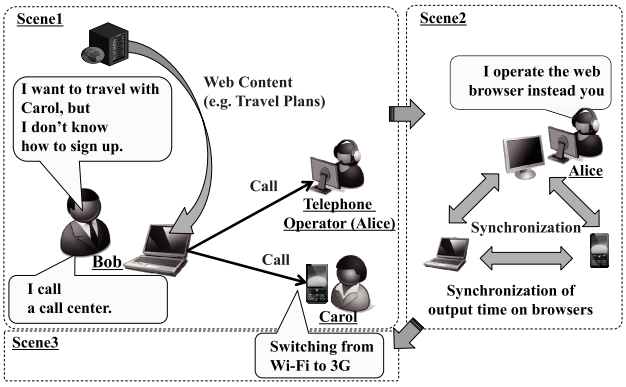

Fig. 1 Use case scenario.

There are several reasons for the occurrence of out-of-synchronization. First, when the bandwidths of the access network of the users are different, the arrival times of the packets may be different. Second, in the case of the mobile phone, the bandwidth of the access network for users differs during the communication. Differences in the arrival times of packets cause differences in the output time. Third, the rendering times are differ-

[1] KDDI R&D Laboratories Inc., Fujimino, Saitama 356–8502, Japan
[a)] ka-tasaka@kddilabs.jp

ent among browsers.

In order to prevent the out-of-synchronization problem, several methods have been studied [4], [5], [6], [7], [8], [9]. These studies synchronize the actual output time of the voice and video data among devices and ensure the quality of the real-time communication. This is realized by sharing the generation time and the estimated output time among devices and by controlling the actual output time according to the network delay.

However, they do not deal with the synchronization of non-streaming content such as web contents. In addition to this issue, previous methods [1], [2], [3], [4], [5], [6], [7], [8], [9] do not focus on the sharing of browser operations among conversational partners.

We propose a web synchronization method (WSM) to share browser operations among browsers of conversational partners and to synchronize the output time of voice, video and web contents. WSM starts sharing of web contents and browser operations among conversational partners who are using voice and video contents. Moreover, WSM synchronizes output times of web contents and the operations. For synchronization, we consider the heterogeneity of not only the network delay but also the rendering performance on browsers and propose a method to control the output timings of webpages by absorbing the performance between devices of users. For considering the feasibility, we also realize a method without additional software or a plugin on a browser.

The rest of this paper is structured as follows. Section 2 describes related works on conventional methods and issues for smooth communication in a heterogeneous environment. Section 3 provides a detailed description of a novel web synchronization method, while Section 4 shows the prototype implementation and performance evaluations. Section 5 summarizes the conclusion.

2. Related Works

This section describes existing methods and issues on continually providing users with an environment for smooth browser-based communications.

2.1 Web Contents Sharing

Several methods for sharing web contents among users have been studied [1], [2], [3]. These methods share the operations of web browsers. Method [1] shares browser operations (webpage movement, webpage scroll, etc.) among browsers including a plugin. This method also manages control authority for sharing of browser operations. Method [2] transmits a browser synchronization signal (e.g., URL of homepage) among devices without preparing a centralized server by using NAT traversal technique. Method [3] considers that users move to a different location and change devices during communication. It keeps web information (tabs, history, forms, etc.) so that the information can be recovered from any device connected to the Internet.

However, existing methods do not focus on sharing among conversational partners (e.g., users and an operator) and synchronization well in a heterogeneous environment such that some users use a PC connected with a fixed network and other users use a smartphone on a mobile network.

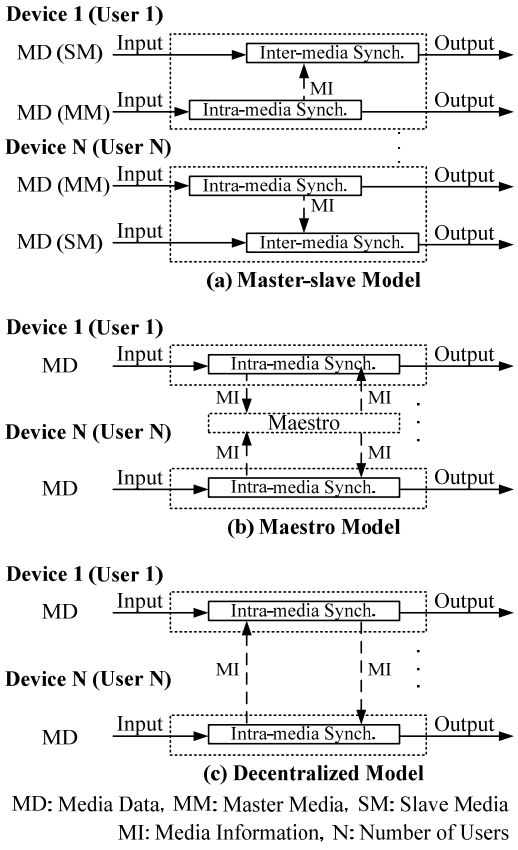

Fig. 2 Typical control models for synchronization among devices.

2.2 Synchronization for Streaming Content

Many researchers have studied methods for synchronization regarding the output times of streaming content such as voice and video [4], [5], [6], [7], [8], [9]. These methods delay the output times of each data packet to ensure the output intervals of the same kinds of media (voice or video) data are equal to the generation intervals of the same data. They achieve intra-media synchronization, which is the difference time between a generation intervals and an output interval is less than the threshold (320 ms) [4], [10].

These methods [4], [5], [6], [7], [8], [9] also achieve an inter-media synchronization that synchronizes the output times between different kinds of streaming data such as voice and video.

Previous methods [1], [2], [3], [4], [5], [6], [7], [8], [9] described above do not focus on the sharing of browser operations among conversational partners. Moreover, methods [4], [5], [6], [7], [8], [9] synchronize the output time of streaming content, but methods [1], [2], [3], [4], [5], [6], [7], [8], [9] cannot synchronize non-streaming content such as web contents.

2.3 Typical Control Model for Synchronization and Our Target

Typical models for conventional synchronization methods are classified into the following three kinds: the Master-slave model (**Fig. 2** (a)) [4], [5], the Maestro model (Fig. 2 (b)) [6], [7] and the Decentralized model (Fig. 2 (c)) [8], [9].

In the Master-slave model (Fig. 2 (a)), the user decides the pri-

ority for media data such as voice and video data. The media data with the highest priority is called the master media, while the other media data is called the slave media. The device that receives master media is defined as the master device, and those that receive slave media are defined as slave devices. Slave devices decide the actual output time of the slave media based on the media information including the generation time and the output time of the master media. The master device distributes the media information to the slave devices. The advantage of this model is to retain the quality of the master media. This model is typically used in TV communication, where voice data is selected as the master media, because it can retain the quality of voice communication over visual communication.

In the Maestro model (Fig. 2 (b)), an entity (maestro) collects the media information of each media data from all devices and decides the actual output times of the consequent data packets. Each device outputs the data packets based on the output times informed by the maestro. This model is typically used in online games since all types of data packets should be treated with the same priority. However, this model tends to increase the output delay compared to the Master-slave model, because each device outputs a data packet after sending the media information to the maestro and receiving the actual output time from the maestro.

In the Decentralized model (Fig. 2 (c)), each device mutually exchanges media information and individually determines the actual output times of each data from the media information. Therefore, in this model, all devices can output data with less delay than the Maestro model. However, this model tends to increase the network load compared to other models because all pairs of devices must frequently exchange media information.

In contrast, our method considers synchronization not only for streaming data but also for non-streaming data of browser based communication. In this communication, our method uses the Maestro model in Fig. 2 (b) and the maestro module controls the output times of the web content for multiple devices in a heterogeneous environment based on the rendering times in addition to network delays. The module needs to decide the timing to send/output web content and share it with the same priority. We describe the details in Section 3.2.

3. Proposed Method: A Web Synchronization Method for Browser-Based Communications

We propose a novel web synchronization method (WSM) that shares browser operations among conversational partners and meets out-of-synchronization without any additional software and a plugin on browsers. We believe that WSM is an essential technology for smooth voice and visual communications in a heterogeneous environment.

3.1 Requirements

We consider the following requirements for synchronization in a heterogeneous environment.

Requirement 1: Starting sharing of browser operations independent of the timing of communication by voice and/or video.

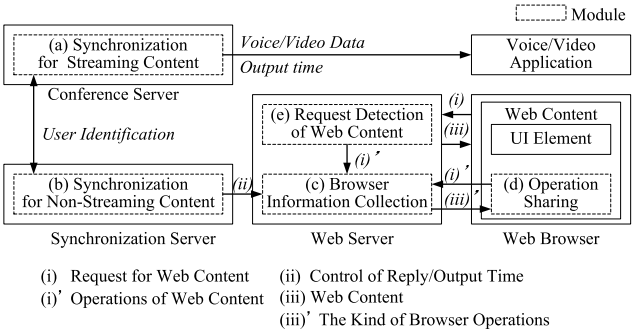

Fig. 3 Overview of architecture as module composition.

At this time, WSM is unnecessary additional software and no plugin is required for feasibility. Users can utilize a communication service by using their devices with web browsers without adding any software or plugin.

Requirement 2: Synchronizing the output times of the web content among conversational partners in a heterogeneous environment. The difference in the output times must be within 320 ms [4], [10]. If the difference is over 320 ms, some users cannot understand the received conversation because all users cannot look at the same webpage at the same time.

Requirement 3: Continuing to synchronize the output times of the web content among web browsers even if a device changes the connected access network, e.g., Wi-Fi to 3G, during the communication.

3.2 Overview of the Proposed Web Synchronization

We show an overview of the architecture with WSM in **Fig. 3**. This architecture is based on the Maestro model because all devices equally share web operations without any additional software on browsers via a web server in a heterogeneous environment.

A synchronization module for streaming (Fig. 3 (a)) is included in the Maestro model. This module controls the output time of voice/video data by changing the timing to send each data and indicating the buffering time according to network delay.

The new module for non-streaming content (web content) (Fig. 3 (b)) plays the role of maestro in the Maestro Model (Fig. 2 (b)). This module decides the timing to send and output web content and shares it with the browser information collection module (Fig. 3 (c)). The browser information collection module collects the operations to be synchronized by scripts (Fig. 3 (d), **Fig. 4**) such as JavaScript on a browser. The request detection module (Fig. 3 (e)) detects a webpage movement as a browser operation.

Section 3.3 shows detailed methods for detecting browser operations and sharing the operations among conversational partners without any additional software for Requirement 1. Section 3.4 shows a method of synchronizing the output times of the web content for Requirement 2. Section 3.5 shows a method of adjusting the output time in the case where the connected access network of a device changes for Requirement 3.

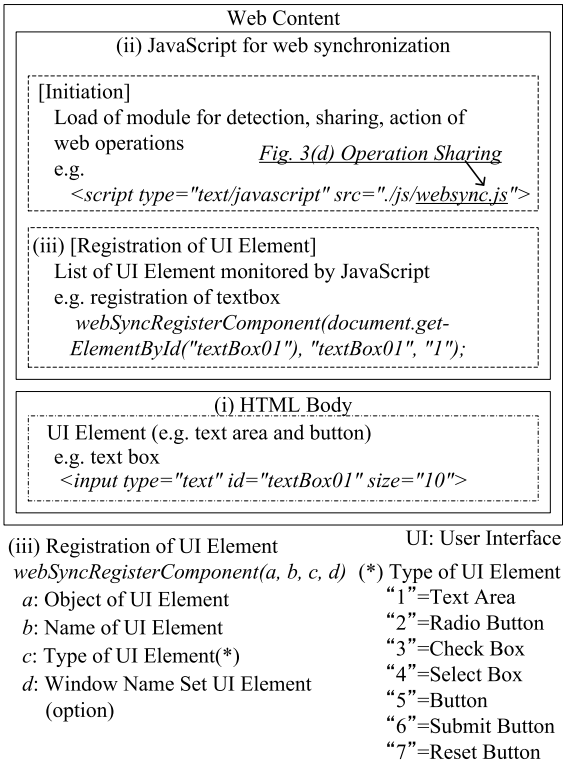

Fig. 4 A method of creating web content for web synchronization.

3.3 Detection and Sharing of Browser Operations

(1) Detection of user operations

The proposed method detects browser operations in modules on a web server (Fig. 3 (e)) or on each browser (Fig. 3 (d)) according to the kind of browser operation.

For synchronization of a webpage movement, a web server detects the operation from a request message of the web content in Fig. 3 (e). The detection in the web server can reduce the time to send a control message to notify browser operations of the web server compared with the detection in each browser.

On the other hand, it is difficult for a web server to detect some operations such as a character input in a text area. In this case, a browser detects the browser operations in Fig. 3 (d) and sends the information to the web server. The method of detecting the browser operations are described in detail as follows.

A service provider (or a proxy) creates a web content including scripts (e.g., JavaScript) that detects and shares browser operations. This script does not force users to install any additional software or any plugin for the web browser.

Figure 4 depicts a method of including a script for detecting and sharing browser operations. Service providers create codes for the user interface (UI) such as a button and a text area on the body part in a web content (Fig. 4 (i)). They set an identification of the UI element (Fig. 4 (iii)) to JavaScript for detecting and sharing browser operations (Fig. 4 (ii)). When a web browser loads the web content, it registers a list of UI elements (target of synchronization) to the synchronization module (Fig. 3 (b)) via the browser information collection (Fig. 3 (c)). If users operate UI elements without registration, the synchronization module does not share the operations. Users can hide the web content including privacy information.

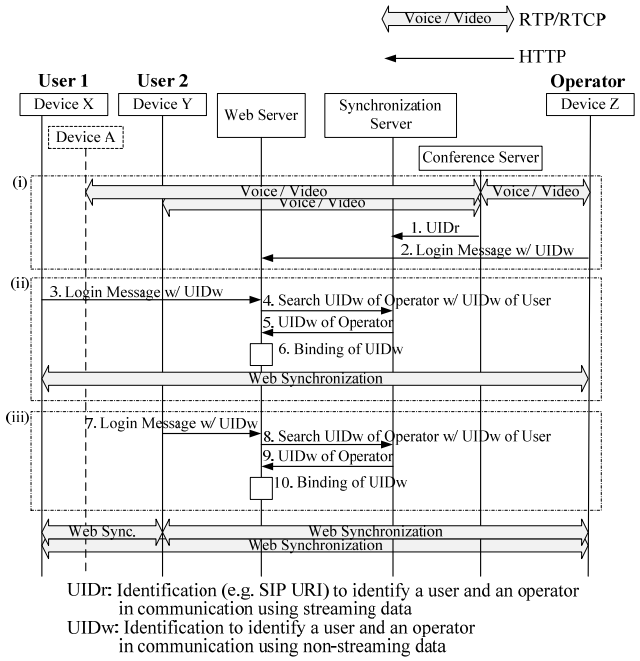

Fig. 5 Sequence of grouping users after starting a voice/video communication.

The service providers can newly add codes for synchronizing browser operations (undefined UI element) such as webpage scroll and drawing action (e.g., line and circle). For new browser operations, the service providers define the argument, *webSyncregister* (e.g., document.getElementById ("scroll"), "scroll", "8") and create codes of JavaScript which detects and acts the operations.

After the registration, when a user operates a UI element on a web content, the script detects the kind of operation. Other browsers load the operation sharing module (Fig. 3 (d)) and run the operation after receiving the kind of browser operation and the value from the web server.

(2) Sequence to start sharing user operation among browsers

When the web server or script described in Fig. 4 detects a browser operation, the proposed method shares it among browsers. For identifying each media data in Fig. 3, the proposed method groups ID (UIDr (e.g., SIP URI)) for a streaming data with ID (UIDw (e.g., Log-in ID)) for non-streaming data.

Figures 5 and **6** show a sequence to group communicating users by using IDs and start a web synchronization.

Case1: Grouping after starting voice/video communication

This is the case where a user asks an operator to supplement voice/video communication with sharing of web content. In this case, users and an operator share browser operations after starting a voice/video communication.

WSM binds the UIDw of users with the UIDw of the conversational partner (e.g., operator). Figure 5 shows a sequence for grouping each UIDw.

(i) Starting voice/video communication (Fig. 5 (1)–(2))

An operator, User 1 and User 2 in Fig. 5 start a voice/video communication. Then, a conference server shares the UIDr with the synchronization server (Fig. 5 (1)). The operator opens his/her browser in advance for web communication (Fig. 5 (2)).

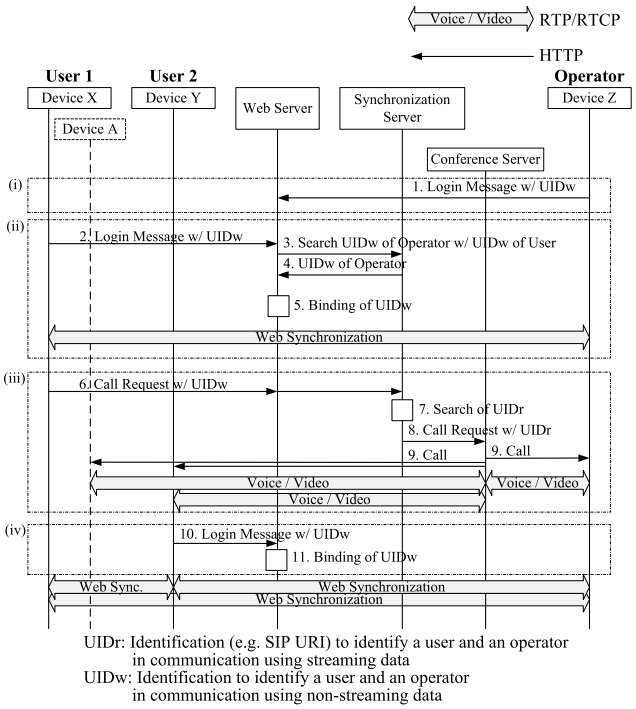

Fig. 6 Sequence of grouping users after starting a web communication.

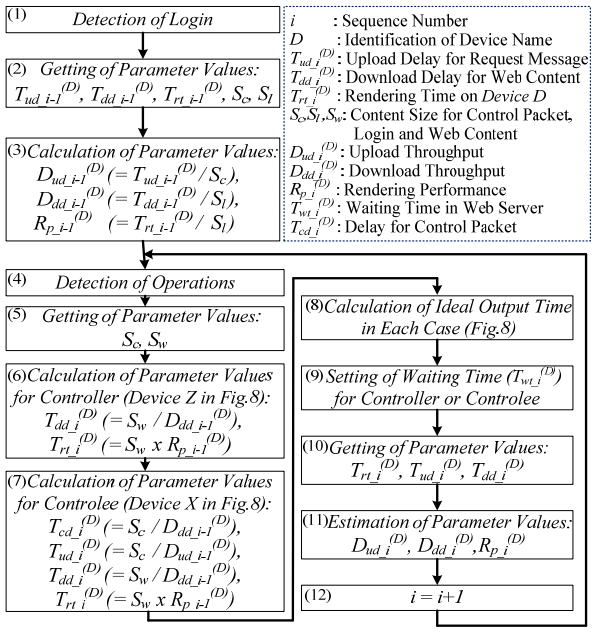

Fig. 7 Flowchart to decide the output time in a synchronization server.

(ii) ID Grouping for web synchronization (Fig. 5 (3)–(6))

User 1 sends a message to login with his/her UIDw from web content such as an online travel site (Fig. 5 (3)). The web server managing web content sends the UIDw of User 1 to the synchronization server (Fig. 5 (4)). The synchronization server searches the UIDr of User 1 from the UIDw and obtains the UIDr of the operator as a conversational partner from the result (UIDr of User 1). In advance, the service provider registers a list of the UIDr and the UIDw with users and operators in a synchronization server.

The synchronization server obtains the UIDw of the operator from the UIDr of the operator and replies the UIDw of the operator to the web server (Fig. 5 (5)). The web server (and the synchronization server) groups each UIDw (Fig. 5 (6)).

(iii) ID Grouping for additional users (Fig. 5 (7)–(10))

Regarding other users such as User 2, devices and servers run the same sequence as that with User 1 and start web synchronization (Fig. 5 (7)–(10)).

Case2: Grouping after starting web communication

Some users call to a call center to ask an operator to supplement details about web content with voice/video communication. Users and the operator start voice/video communication after starting browsing.

WSM searches the UIDw of the operator who can communicate with the user and groups the UIDw. WSM searches the UIDr from their UIDw and calls from a conference server to the UIDr. Figure 6 shows the sequence.

(i) Ready for communication (Fig. 6 (1))

As the ready phase, the operator waits for a call from the user and opens his/her browser in advance (Fig. 6 (1)).

(ii) ID grouping for web synchronization (Fig. 6 (2)–(5))

User 1 sends a message to login with his/her UIDw from the web content to the web server during web browsing (Fig. 6 (2)). The web server sends the UIDw of User 1 to the synchronization server (Fig. 6 (3)) and obtains the UIDw of the operator who can communicate with the user (Fig. 6 (4)). At this time, the web server (and the synchronization server) binds and groups each UIDw (Fig. 6 (5)) for web synchronization.

(iii) Starting voice/video communication (Fig. 6 (6)–(9))

User 1 sends a request message (including the UIDw or UIDr of User 2) for starting a voice/video communication from contact information on his own web site to a web server. The web server forwards it to a synchronization server (Fig. 6 (6)). The synchronization server searches the UIDr of User 1 and User2 and the operator from each UIDw (Fig. 6 (7)) and sends them to the conference server (Fig. 6 (8)).

The conference server calls the users and the operator based on 3rd party call control protocol [11] (Fig. 6 (9)). The users and the operator start a voice/video communication.

(iv) ID grouping for additional users (Fig. 6 (10)–(11))

The web server groups each UIDw (Fig. 6 (10)–(11)) after login by User 2 as well as User 1.

In each case, the processing time on each server is out of scope. We assume the difference of the processing time for each browser is negligible. If that difference increases and causes out-of-synchronization, we need consider load sharing for each server.

3.4 Control of Output Time on the Web Browser

WSM estimates the actual output time of web content based on parameters such as the network delay (throughput) on access networks and the rendering performance of each web browser. **Figure 7** shows a flowchart to estimate the ideal output time. The ideal output time means a time when each device should output each data for synchronization.

(i) Getting initial parameter values (Fig. 7 (1)–(3))

First of all, when the operator and each user login to a web server (Fig. 7 (1)), a synchronization server obtains three initial parameter values to estimate the ideal output time to synchronize

output times (Fig. 7 (2)). The method is described in detail as follows.

When the operator or users login (Fig. 5 (2), (3), Fig. 6 (1), (2)), browsers obtain three initial parameter values: the upload delay ($T_{ud\_i-1}^{(D)}$) of a login message for login, the download delay ($T_{dd\_i-1}^{(D)}$) of web content and the rendering time ($T_{rt\_i-1}^{(D)}$). *D* and *i* mean the identification of the device name and the sequence number of the packet, respectively. The browsers send each value to the web server.

In Fig. 7 (3), the synchronization server calculates the upload throughput ($D_{ud\_i-1}^{(D)}$ (= $T_{ud\_i-1}^{(D)}$ / S_c)) and the download throughput ($D_{dd\_i-1}^{(D)}$ (= $T_{dd\_i-1}^{(D)}$ / S_l)) from the upload delay, the download delay, the content size of the control packet (Sc) and the content size of the login page (S_l). The server calculates the rendering performance ($R_{p\_i-1}^{(D)}$ (= $T_{rt\_i-1}^{(D)}$ / S_l)) from the rendering time and the content size. This function uses these parameters for calculations in Fig. 7 (6) and Fig. 7 (8).

(ii) Calculation of each parameter value (Fig. 7 (4)–(7))

The synchronization server detects the operations (e.g., webpage movement) by receiving a notification from the web server (Fig. 7 (4)). The synchronization server obtains the content size of the control packet (Sc) and the web content (Sw) from the web server (Fig. 7 (5)). The synchronization server calculates the download delay ($T_{dd\_i}^{(D)}$ (= Sw / $D_{dd\_i-1}^{(D)}$)) of the web content and the rendering time ($T_{rt\_i}^{(D)}$ (= $R_{p\_i-1}^{(D)}$ x Sw)) for the controller (Fig. 7 (6)). In the controlee, the server also calculates the download delay ($T_{cd\_i}^{(D)}$ (= Sc / $D_{dd\_i-1}^{(D)}$)) of control packet for sharing the operations and the upload delay of a request message of the web content ($T_{ud\_i}^{(D)}$ (= Sc / $D_{ud\_i-1}^{(D)}$)) (Fig. 7 (7)).

(iii) Setting of ideal output time and waiting time to control output time (Fig. 7 (8)–(9))

The synchronization server calculates the ideal output time based on each value ($T_{cd\_i}^{(D)}$, $T_{ud\_i}^{(D)}$, $T_{dd\_i}^{(D)}$, $T_{rt\_i}^{(D)}$). The ideal output time is the last time as the result of comparison between the estimated output time of the controller ($T_{dd\_i}^{(D)} + T_{rt\_i}^{(D)}$) and that of each controlee ($T_{cd\_i}^{(D)} + T_{ud\_i}^{(D)} + T_{dd\_i}^{(D)} + T_{rt\_i}^{(D)}$) (Fig. 7 (8), **Fig. 8**). The synchronization server decides the waiting time ($T_{wt\_i}^{(D)}$) in such a way that the actual output time becomes equal to the ideal output time (Fig. 7 (9), Fig. 8) within a setting value for time out of web sessions on browsers.

Case 1 (Fig. 8 (a)): The web server sets the waiting time ($T_{wt\_i}^{(z)}$) for the controller and delays the sending of the web content to the controller if inequality (1) is established (e.g., Controller is Device Z and Controlee is Device X).

$$T_{dd\_i}^{(z)} + T_{rt\_i}^{(z)} < T_{cd\_i}^{(x)} + T_{ud\_i}^{(x)} + T_{dd\_i}^{(x)} + T_{rt\_i}^{(x)} \quad (1)$$

Case 2 (Fig. 8 (b)): The web server sets the waiting time ($T_{wt\_i}^{(x)}$) for the controlee and delays the sending of the web content if Eq. (2) is established.

$$T_{dd\_i}^{(z)} + T_{rt\_i}^{(z)} > T_{cd\_i}^{(x)} + T_{ud\_i}^{(x)} + T_{dd\_i}^{(x)} + T_{rt\_i}^{(x)} \quad (2)$$

The synchronization server sends a control packet that requests the operations from the controller to each controlee via the web server. Each controlee receives the control packet and runs the operation (e.g., webpage movement). The web server sets the waiting time for the controlee. After the waiting time, the web server sends the web content to the controlee.

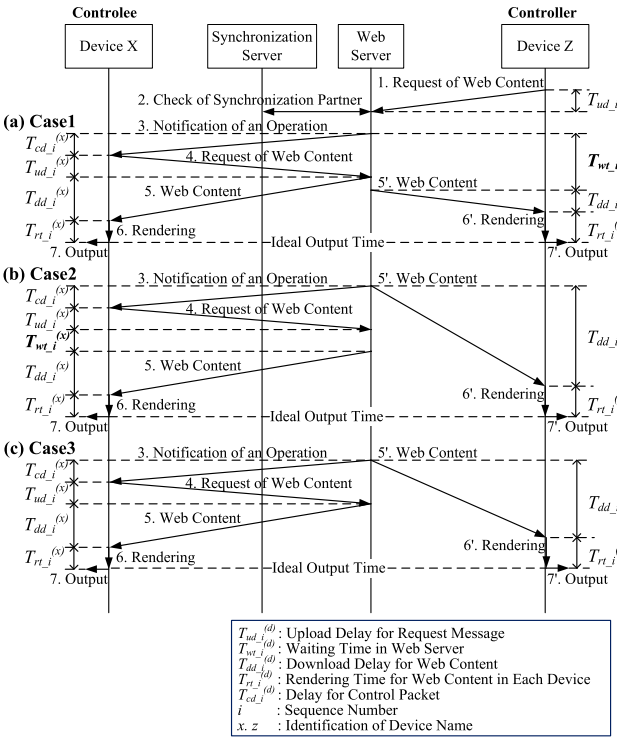

Fig. 8 Ideal output time and setting of waiting time in each case.

Case 3 (Fig. 8 (c)): There is no waiting time for all devices if Eq. (3) is established.

$$T_{dd\_i}^{(z)} + T_{rt\_i}^{(z)} = T_{cd\_i}^{(x)} + T_{ud\_i}^{(x)} + T_{dd\_i}^{(x)} + T_{rt\_i}^{(x)} \quad (3)$$

(iv) Updating each parameter value (Fig. 7 (10)-(12))

The web server sends web content to each device. The synchronization server obtains each parameter value from each device via the web server in the same sequence for initial parameter values (Fig. 7 (10)). The synchronization server updates the throughput, the rendering performance (Fig. 7 (11)) and the sequence number (Fig. 7 (12)).

3.5 Adjustment of Output Time

Our proposed method adjusts the output times of web content for each device so that the web synchronization works well even If the devices are connected to different access networks (e.g., Wi-Fi area and mobile phone area) according to the user's movement. In this time, the browser re-connects web sessions to control the browser with the server. The browser runs the sequence based on Ajax/Comet.

When a device changes the access network, three parameter values ($T_{cd\_i}^{(D)}$, $T_{ud\_i}^{(D)}$, $T_{dd\_i}^{(D)}$) have the possibility of fluctuating at either of the following points (Point 1: before notifying the browser operations of each controlee, Point 2: between point 1 and the point before requesting for a new webpage from each controlee, Point 3: between point 2 and before sending a new webpage). At each point, the synchronization server estimates the network delay and the rendering time and sets an appropriate waiting time. $T_{cd\_i}^{(D)}$ in Point 1, $T_{ud\_i}^{(D)}$ in Point 2 and $T_{dd\_i}^{(D)}$ in Point 3 change according to the throughput of a new access network. The synchronization server updates or sets $T_{wt\_i}^{(D)}$ ac-

cording to the change in $T_{cd\_i}^{(D)}$, $T_{ud\_i}^{(D)}$ and $T_{dd\_i}^{(D)}$.

If the synchronization server estimates that the arrival of web content to the controller is earlier than that to the controlee as shown in Case 1, it sets the waiting time $T_{wt\_i}^{(D)}$ for the controller. If the synchronization server estimates that the arrival of a web content to the controlee is earlier than that to the controller as shown in Case 2, it sets $T_{wt\_i}^{(D)}$ for the controlee. Otherwise if the arrival times are the same as shown in Case 3, the waiting time is not set for the controller or the controlee.

4. Implementation and Performance Evaluation

4.1 Implementation of Proposed Method

We implemented modules as described in Section 3 and installed the modules onto a web server, a synchronization server and a conference server (OS: Redhat Enterprise 5 64 bit, CPU: Xeon 3.0 x4, Memory: 4 GByte).

We used asterisk [12] as a tool for the conference server. For web synchronization, we used Ajax (Asynchronous JavaScript and XML) as a tool to send the browser operation to a web server and Comet (Reverse Ajax) as a tool to receive the browser operation from the web server. To start and control voice and video communications using Real-Time Transport Protocol (RTP) [13], we used Session Initiation Protocol (SIP) [14]. We used Simple Object Access Protocol (SOAP) for sending/receiving messages between the web server and the synchronization server, and between the synchronization server and the conference server.

Figure 9 shows an experimental environment of the prototype system. Users 1, User 2 and Operator make voice communication using Device A (a mobile device), Device Y (a mobile device) and Device Z (a fixed device), respectively. For a non-streaming communication, they use Device X (a fixed device) installed with Firefox browser, Device Y installed with Android standard browser and Device Z installed with Internet Explorer, respectively.

Each fixed device has a LAN interface (100M-TX). Device X and Z are connected to Gateway 1 and 3, respectively. On the other hand, each mobile device has a Wireless LAN (WLAN) interface (IEEE802.11g) connected to each Wi-Fi AP (Access Point). Device A and device Y access to Gateway 1 and 2, respectively. Device Y also has a communication interface for access to a Mobile Phone Network.

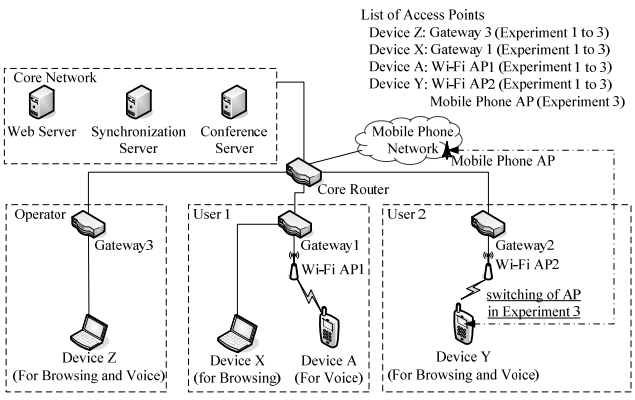

Fig. 9 Experimental environment.

The synchronization server and the web server connect to the core router via the core network. Each gateway and core router has a function of a network emulator (NIST Net [15]) and is capable of changing the network bandwidth for performance evaluation. The NIST Net network emulator is a general-purpose tool for emulating performance dynamics in IP networks.

4.2 List of Experiments

4.2.1 Performance of Web Synchronization

We measured the time difference in the output of web content and operation among user's browsers using the proposed method and the conventional method described in Section 2.1 in order to compare the accuracy of web synchronization. The web server collects the output time from each browser and calculates the difference as the difference time by subtracting the output time. In the measurement, we changed the setting of two parameters (the throughput (network delay) and the content size (rendering time)). We evaluated the influence of the changes in the network bandwidth and web content size on the performance of web synchronization.

Experiment 1: We changed the bandwidth on the access network (WLAN) for Device Y in Fig. 9 from 1 Mbps to 20 Mbps (actual max speed in Wi-Fi AP) by setting the bandwidth in Gateway 2. We fixed the bandwidth of other access networks (Device X and Device Z: 20 Mbps) and the web content size (3 Mbyte: size of the top page in a Japanese travel agency).

We evaluated the influence of the change in throughput on web synchronization by using the results of this measurement.

Experiment 2: We changed the content size on the web server from 200 KByte to 3 MByte and fixed the throughput (Device X and Device Z: 20 Mbps, Device Y: 1 Mbps) on access networks. Device Y accesses to Wi-Fi AP in this measurement.

We evaluated the influence of the change in the content size on web synchronization by using results of this measurement.

4.2.2 Performance of Synchronization Adjustment

Experiment 3: We measured the output time of the web content on each device when Device Y switches to a different access network between a WLAN and a mobile phone network. In this case, we fixed the bandwidth on access networks for Device X and Device Z and the content size (3 Mbyte).

We evaluated the adjustment of web synchronization when a device changes access networks.

4.2.3 Scalability

Experiment 4: We measured the maximum number of web sessions that can be simultaneously handled in the architecture of WSM (Fig. 3). In this experiment, we increase the number of web sessions from 3 to 1,000 by simulation.

From this measurement result, we show the influence of the change in the number of web sessions on web synchronization.

4.3 Experimental Results

4.3.1 Performance of Web Synchronization

Figures 10 and **11** indicate the experimental results corresponding to Experiments 1 and 2.

Measurement Result 1: Figure 10 shows the results of Experiment 1, which is the maximum difference in the output time of

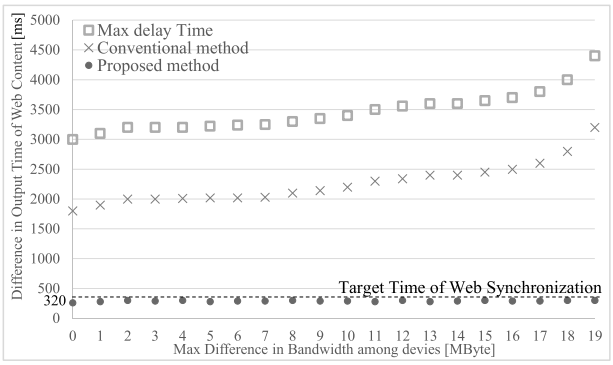

Fig. 10 Difference in output time of web content among browsers according to increase of difference in bandwidth among devices.

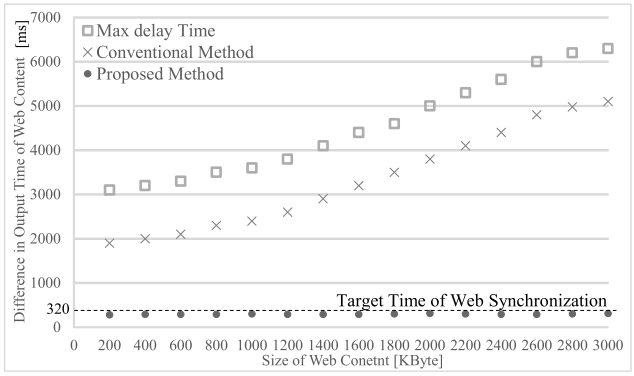

Fig. 11 Difference in output time of web content among browsers according to increase of web content size.

web content among browsers of each user and operator according to the difference in throughput.

The maximum delay time in Fig. 10 specifies the delay from the time when the web content is requested at the controller to the time when the web content is output to the last controlee.

In the proposed method, the difference in the output time is from about 200 ms to about 300 ms regardless of the difference in bandwidth (throughput) among devices. These results mean that the difference between actual values of the network delay, the rendering time and their estimated values is within the target time of web synchronization by the proposed method described in Section 3.4. This indicates that the proposed method achieves synchronization for web communication within the target time (320 ms).

On the other hand, in the conventional method, the difference in the output times increases according to the difference in bandwidths among devices. This indicates the conventional method depends on the network delay and is out-of-synchronization.

The above results indicate that the proposed method can continually synchronize the output time on each browser even if the difference in network delay and web content increases.

Measurement Result 2: Figure 11 shows the results of Experiment 2. The difference in the output times in the proposed method are from about 200 ms to about 300 ms even if the differences in content size increase. These results mean that the difference between actual values of the network delay, the rendering time and their estimated values is within the target time of web synchronization by the proposed method described in Section 3.5. This indicates that the proposed method achieves synchronization for

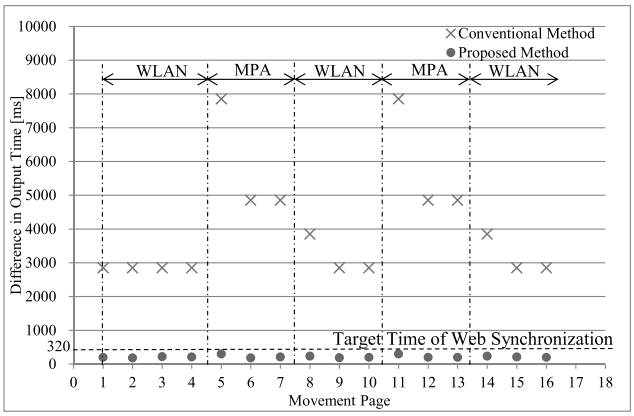

Fig. 12 The difference in output time of web content when a device moves between WLAN and mobile phone area.

web communication within the target time (320 ms).

On the other hand, in the conventional method, the differences in the output times increase according to the difference in the content size on the web server. This indicates that the state in the conventional method is out-of-synchronization and dependent on the content size and rendering time.

Therefore, the proposed method achieves a smooth communication from continual synchronization even if the content size and the difference in the rendering time among browsers increase.

The results of Experiments 1 and 2 indicated that the proposed method can synchronize the output times of non-streaming content among browsers within 320 ms in heterogeneous environments to fulfill Requirement 2 in Section 3.1.

4.3.2 Performance of Synchronization Adjustment

This section indicates the measurement results on the change in the difference in output time when Device Y in Fig. 9 moves between WLAN and mobile phone areas. We evaluated the adjustment of synchronization from the results.

Measurement Result 3: **Figure 12** shows the results of Experiment 3, which is the change in the differences in the output times on devices.

In the proposed method, the difference in the output times was in the range between about 200 ms and 300 ms even if network delay (or throughput) changed according to a change in access networks. This result indicates that the proposed method achieves synchronization for web communications within the target time (320 ms).

In the conventional method, the difference in the output times fluctuates according to the change of access network by Device Y. In downloading of the 5th, 8th, 11st, 14th webpage after the changes, the difference in output times increases due to time to switch an access network. The difference in the output times increases when the link speed of the connected access network for a device changes from a higher one to the lower one. This result indicates that the difference in throughput among devices influences the output times in the conventional method.

Therefore, the proposed method continually achieves web synchronization of the output time among browsers even if the throughput of the access network changes to fulfill Requirement 3 in Section 3.1.

4.3.3 Scalability

Measurement Result 4: We confirmed that our system could handle all sessions and synchronize output times even if the number of web sessions was increased from 3 to 1,000. The number of web sessions corresponds to the number of users in the experiment.

The results of Experiment 4 indicates that the proposed method can stably synchronize the output time. If service providers want to handle sessions of over 1,000 users and a server cannot handle all web sessions at the same time, they can take measures such scale out and scale up of each server based on existing technologies. As a future work, we will consider efficient management of web sessions.

5. Conclusion

We propose a web synchronization method (WSM) that starts sharing browser operations among browsers before/after the start of voice and video communications. WSM meets issues on out-of-synchronization of conventional methods in a heterogeneous environment where the network delay and rendering time fluctuate. WSM also continues to synchronize the output times of non-streaming data in addition to the output times of voice and video data even if a device is connected to different access networks during a conversation. The synchronization is realized by dynamically changing the time to share browser operations and the time to send web content according to the network delay and rendering time on each browser. We implemented a prototype system and measured the difference in the output time among browsers. The experimental results show that WSM achieves web synchronization within about 300 ms while the target time is 320 ms even if the network delay and the rendering time fluctuate. From the experimental results, users can receive the same conversation content as the web content on browsers at the same time. Moreover, the proposed method is applicable to a training service and teleconference for business as scenarios that require synchronization. In the training service, a teacher remotely instructs his/her students on how to show commercial products to their customer. In the teleconference service, business partners remotely share and edit materials for a conference on their browsers. In future, we will consider web synchronization according to the processing load on devices and servers and evaluate the availability.

Acknowledgments We are indebted to Dr. Yasuyuki Nakajma, President and Chief Executive Officer of KDDI R&D Laboratories Inc. and Shigehiro Ano, Executive Director, for their continuous encouragement of this research.

References

[1] Hwan-Gu, L., Won-Tae, K., Sun-Ja, K. and Cheol-Hoon, L.: Design and Implementation of Mobile Cobrowsing Service which supports the sharing of webpage among mobile users, *Proc. IEEE Conf. Convergence and Hybrid Informatin Technology* (*ICHIT'08*), pp.266–269 (2008).

[2] Hongo, N., Yamamoto, H. and Yamazaki, K.: Browser Synchronization System for Supporting Elderly People and IT Shortfalls, *Proc. IEEE Conf. Computer Software and Applications Conference* (*COMPSAC 2013*) (2013).

[3] Valle, R., Passito, A., Novellino, R. and Penaranda, A.: Synchronization Web Browsing Data with Browserver, *Proc. IEEE Symp. Computers and Communications* (*ISCC 2010*), pp.738–743 (2010).

[4] Ishibashi, Y. and Tasaka, S.: A Media Synchronization Mechanism for Live Media and Its Measured Performance, *IEICE Trans. Commun.*, Vol.E81-B, No.10, pp.1840–1849 (1998).

[5] Wongwirat, O. and Ohara, S.: Haptic media synchronization for remote surgery through simulation, *IEEE Trans. Multimedia*, Vol.13, No.3, pp.62–69 (2006).

[6] Wagner, P. and Frossard, P.: Playback delay and buffering optimization in scalable video broadcasting, *Proc. IEEE Conf. Multimedia Services Access Network* (*MSAN'05*), pp.746–752 (2005).

[7] Ishibashi, Y. and Tasaka, S.: A group synchronization mechanism for live media in multicast communications, *Proc. IEEE Conf. Global Communications* (*Globecom'97*), pp.746–752.

[8] Ishibashi, Y. and Tasaka, S.: A distributed control scheme for causality and media synchronization in network multimedia games, *Proc. IEEE Conf. Computing Communication and Network* (*ICCCN 2002*), pp.144–149 (2002).

[9] Tasaka, K., Imai, N., Isomura, M. and Idoue, A.: A media synchronization method for real-time group communication in a multiple device environment, *Proc. IEEE Conf. Intelligence in Next Generation Networks* (*IMSA 2008*), pp.1–6 (2008).

[10] ITU-T Recommendation G.114, One-way transmission time (2003).

[11] Rosenberg, J., Peterson, J., Schulzrinne, H. and Camarillo, G.: Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP), RFC3725 (2004).

[12] Digium, Asterisk: The open source PBX and telephony platform (2006).

[13] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: A Transport Protocol for Real-Time Applications, RFC1889 (1996).

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

[15] NIST Net, available from ⟨http://www-x.antd.nist.gov/nistnet/⟩.

Kazuyuki Tasaka was received his B.E. degree from Niihama National College of Technology in 2002, and his M.E. and Ph.D. degrees from Nara Institute of Science and Technology in 2004 and 2010, respectively. Since joining KDDI in 2004, he has worked in the field of network architecture, communication protocols and mobile communications. He is currently a research engineer of the Smart Network Administration Lab. in KDDI R&D Laboratories, Inc. He received the Best Paper Award for Young Researcher of IPSJ National Convention in 2005, the FIT Young Researcher Award of IPSJ National Convention in 2009 and the Paper Award of Multimedia, Distributed Cooperative and Mobile Symposium in 2007 and 2011. He is a member of IPSJ and IEICE.

Tomohiko Ogishi was received his B.E. degree in electrical engineering from the University of Tokyo, Japan, in 1992, and Ph.D. degree in engineering from University of Electro Communications, Japan, in 2008, respectively. Since joining KDD in 1992, he has been working in the field of Testing and monitoring of communication systems. He is currently the Senior Manager of Smart Network Administration Lab. in KDDI R&D Labs Inc. He received Young Engineer Award of IEICE in 1998.