Vol.2015-OS-135 No.8
Vol.2015-EMB-39 No.8
2015/11/24

分散型 Web ブラウザにおける遠隔会議システムの実装

談　エン[†1]　新城靖[†1]　李　咏[†1]　佐藤　聡[†1]　中井　央[†1]

概要：Facebook や Twitter などの集中型 SNS(Social Networking Service) は中央サーバを用いて実装されている。そのためプライバシ侵害等の問題が指摘されている。この論文はこの問題を解決するための、分散型 Web ブラウザにおける遠隔会議システムの実装について述べている。分散型 Web ブラウザは中央サーバを用いることなくブラウザ間通信を可能にする、Web 上の分散型 SNS アプリケーションを実行する基盤である。この論文で述べている遠隔会議システムは、ノード間で WebRTC によりビデオ配信やファイル共有機能を提供する。この遠隔会議システムの実現を通じて、分散型 Web ブラウザにおけるアプリケーションの実現方法を明らかにしている。

キーワード：分散型ソーシャルネットワーキングサービス、遠隔会議システム、分散型 Web ブラウザ

1. はじめに

Social Networking Service(SNS) はコミュニケーションツールとして普及している。その多くは中央サーバを用いて実装されており、プライバシ侵害等の問題がある。例えば、2013 年 6 月にはアメリカ政府 NSA(National Security Agency) が Microsoft 社や Facebook 社等から個人情報を収集していた事実が判明し、社会問題になっている [4]。このようなプライバシの問題に対処するために分散型 SNS が提案されている。分散型 SNS とは、中央サーバを用いることなく、ソーシャルアプリケーションを利用するため仕組みである [1][13]。

現在、インターネットの普及と通信環境の改善にともなって多くの人は遠隔会議システムをよく利用するようになってきた。遠隔会議システムを利用すると、参加者にビデオを配信したり、ファイルを転送したり、テキストの送信を行うことができる。これらの遠隔会議システムは、Web ブラウザで動作するものが多い。このような代表的なサービスとしては Ustream、YouTube Live および Google Hangouts があげられる。オープンソースのアプリケーションとしては、OpenMeetings がある。それらの多くのは中央サーバを用いて実現されている。集中型 SNS と同様に中央サーバを用いて遠隔会議システムを実装すると、それらを監視されたら、簡単にプライバシが侵害される。

[†1] 筑波大学
Tsukuba Uniersity

このような中央サーバに関する問題を解決するために、本研究の目的を中央サーバを使わないで、分散型 SNS のメンバを対象とした遠隔会議システムを実現することと定める。本遠隔会議システムは、Web ブラウザで動作する他のアプリケーションと協調して動作するようにする。この目的を達成するために、本研究では、分散型 Web ブラウザ [12] 上で動作するアプリケーションとして遠隔会議システムを実装する。ビデオデータを転送するために WebRTC[16] を用いることにする。

2. 分散型 Web ブラウザ

従来のブラウザ間の通信は 3 章で述べる WebRTC を除いて中央サーバを経由してなされる。本研究室では中央サーバに依存する問題を解決するため、分散型 Web ブラウザを開発している [12]。分散型 Web ブラウザは、分散型 SNS のアプリケーションを実行するための基盤、すなわち分散型オペレーティングシステムの一種と見ることもできる。協調アプリケーションを開発する際、分散型 Web ブラウザの機能を利用すれば、通信部分を考える必要がないため、開発の手間を減らすことができる。また、ブラウザ間の通信は中央サーバを経由しないので、機密性を満たすこともできる。ストレージとしては各ユーザの PC 上のファイルを用いる。

分散型 Web ブラウザ間の通信機能としては、次のようなものがある。いずれの仕組みも、通信相手を SNS メンバの名前で指定し、Web ブラウザ間で直接通信することを可能にする。

- SkypeRPC[12]。通信路として Skype の AP2AP (Ap-

plication to Apllicatin) 通信を用いる。

- FriendSocket[5]。通信路として、(独自に実行した) XMPP(Extensible Messaging and Presence Protocol) サーバを用いる。
- Social SoftEtherVPN [6]。通信路として、集中型 SNS アカウントで認証された SoftEther VPN を用いる [9]。
- SocialSocket[8]。通信路として、XMPP や WebRTC の基盤技術を用いる。

分散型 Web ブラウザ上で動作するアプリケーションとしては、次のようなものが開発された。

- コメント共有 [10]。ノード間で RPC(Remote Proceduce Call) によりコメントのデータを共有する。
- 協調動画視聴 [5]。グループ通信を実現する通信ミドルウェアを利用して複数人で動画を同時に視聴する機能を持つ。動画のソースは Youtube 等のサーバにある。
- 簡単な協調プラウジング [12]。あるブラウザで開いているページを、他のブラウザで開く。cookie を転送することで、保護されたページも表示できる。

本研究では、分散型 Web ブラウザ上で動作するアプリケーションとして、遠隔会議システムを実現する。本遠隔会議システムでは、動画のソースとして会議に参加しているメンバの PC が備えているカメラから得られたものを利用する。

3. WebRTC

WebRTC[16](Web Real-Time Communication) とは World Wide Web Consortium (W3C) が提唱するリアルタイム通信用の API およびプロトコルである。この API を実装した Web ブラウザではプラグイン無しでボイスチャット、ビデオチャット、ファイル共有のアプリケーションを実行できる。現在、Google Chrome や Firefox 等の主要な Web ブラウザは WebRTC に対応している。それを利用すると、中央サーバを用いることなく、Web ブラウザ間でカメラ、マイク、ファイルから得られたデータをメディアストリームとて送受信できる。ただし、WebRTC は 1 対 1 の通信しかできない。本研究では WebRTC を利用して 3 名以上のメンバが参加できる遠隔会議システムを実装する。

3.1 WebRTC におけるセッション確立の手順

WebRTC を利用しブラウザ間で通信路を確立する時、通信する 2 つのノードは Offer と Answer に分かれる。通信路を確立するために図 1 のようにメッセージを交換する。

(1) Offer と Answer は、それぞれ STUN (Session Traversal Utilities for NATs) サーバに要求を送る。STUN サーバは、Offer および Answer にそれぞれ IP アドレスと UDP ポート番 号を返す。
(2) Offer は自分の SDP 情報を Answer に送信する。SDP (Session Description Protocol) は P2P(Peer-to-Peer) で接続するメディアの種類（音声、ビデオ)、自身の IP アドレスや UDP ポート番号等を記した文字列である。
(3) Offer は自分の SDP 情報を Answer に送信する。
(4) Offer は自分の ICE 情報を Answer に送信する。ICE (Interactive Connectivity Establishment) は P2P で接続するブラウザの通信経路の情報を示した文字列である。
(5) Answer は自分の ICE 情報を Offer に送信する。

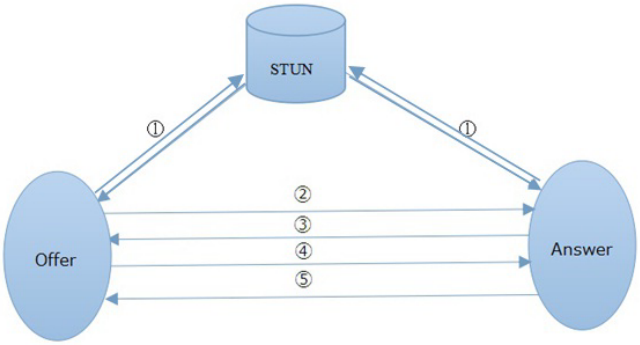

図 1 WebRTC におけるセッションの手順

本遠隔会議システムは分散型 Web ブラウザが提供する通信機能を利用し、Offer と Answer の間で SDP 情報と ICE 情報を交換する。

3.2 PeerConnection と DataChannel

PeerConnection は 2 つの Web ブラウザ間でマルチメディアセッションを確立するための API である。セッションを確立する時に指定するパラメタには、コーデック、暗号化手法、帯域管理等機能を含む。表 1 は Web ブラウザ Google Chrome の PeerConnection の API である。PeerConnection API でブラウザの間セッションを確立した後、Datachennel API を使ってデータ送受信するための通信路を Web ブラウザの間で確立することができる。

4. 遠隔会議システムの設計

本遠隔会議システムを 1 人または少数のユーザが同時にビデオを配信できることを目標に設計する。現在代表的な遠隔会議システムとして Ustream、YouTube Live があげられる。それらの多くは中央サーバを用いて実現されている。集中型 SNS と同様に中央サーバを利用すると、それらを監視されたら簡単にプライバシが侵害される。

本遠隔会議システムは、分散型 Web ブラウザ上で動作するアプリケーションとして実装する。ビデオの配信に WebRTC を利用するので、中央サーバを用いる方法と比べて不当な監視は難しくなる。

表 1　PeerConnection API

関数名	説明
PeerConnection(config)	PeerConnection を作る関数である。引数の config では、STUN サーバ、ICE 情報等を指定する。
addStream(MediaStream)	PeerConnection に MediaStream を追加する。MediaStream とは、音声やビデオを表現しているストリームである。たとえば、カメラから動画像を表すストリームは、getUserMedia() 関数で作成することができる。

4.1　遠隔会議システムが提供する機能

本会議システムでは 1 人または少数のユーザが交代で同時にビデオを配信できるようにする。その他の参加者は、ビデオを受信する。分散型 SNS で利用することを想定し、参加者数が 100 人、その中で最大 5 人が交代で同時にビデオを発信できることを目標とする。本遠隔会議システムの機能として、ビデオの配信に加えて、会議資料の配付のためのファイル共有およびテキスト・チャット機能も実現する。

4.1.1　ビデオ配信

図 2 にビデオ配信機能のユーザインタフェースを示す。会議を主催するユーザが StartVideo ボタンを押すと、そのユーザのカメラのビデオが Web ブラウザに表示される。会議に参加するユーザが join ボタンを押し、主催者を指定すると会議に参加することができる。本遠隔会議システムは HTML5 の`<video>` タグを使ってカメラからビデオデータを取得し、WebRTC の PeerConnection API を使って送信する。

4.1.2　ファイル共有

図 2 にファイル共有機能のユーザインタフェースを示す。会議を主催するユーザ、まだは会議に参加するユーザはファイルを送信したい場合、まず Choose File ボタンを押す。すると、ファイル選択するダイアログが表示される。ユーザは、ファイルを選択したら、send ボタンを押す。本遠隔会議システムは、ファイルを WebRTC の Datachannel API を使って送信する。

4.1.3　テキスト・チャット

図 2 にテキスト・チャット機能のユーザインタフェースを示す。会議を主催するユーザ、まだは会議に参加するユーザは、送信したいメッセージをテキストボックスに入れ、sendmessage ボタンを押す。本遠隔会議システムは、メッセージを WebRTC の Datachannel API で送信する。

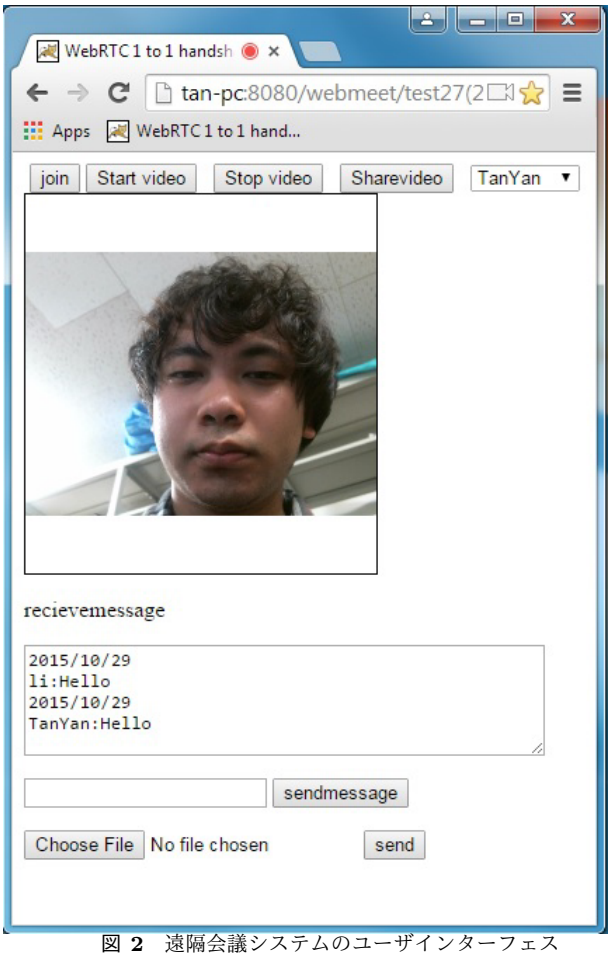

図 2　遠隔会議システムのユーザインターフェス

4.2　木構造を用いた通信形態

Ustream 等の集中型の遠隔会議システムを単純に分散型 Web ブラウザで実装しようとすると、通信経路は図 3 に示したような 1 対 n になる。したがって会議の参加人数はユーザ A が直接対応できる人数に限定されてしまう。WebRCT で標準的なビデオを配信する時約 1.7Mbps の帯域を利用する。例えば 20Mbps の家庭用ネットワークでは

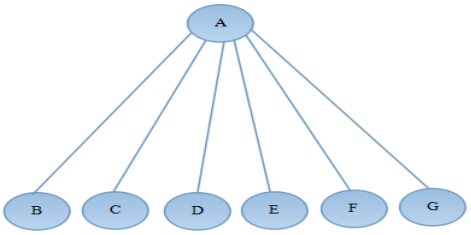

図 3　集中型の通信形態

最大約 10 ユーザしか対応できない。

この問題を解決するため、本研究で実装する遠隔会議システムでは、図 4 のような木構造を用いる。そうすると、ユーザ A は少数の子ノードだけに対応すれば済むことになる。他のユーザは例えばユーザE の場合は、ユーザ B

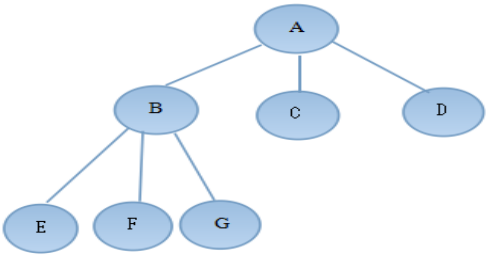

図 4　木構造を用いた通信形態

を経由して、ユーザ A と通信する。そうすると、ユーザ A が直接対応できる人よりも多くの人にビデオを配信できるようになる。

4.3　ビデオ配信の実装

図 5 に、WebRTC を使ったビデオ配信の実装の概略を示す。この図では、中継ノード (Relay node) が親ノード (Parent) からのビデオデータを子ノード (Child) へ中継している。この図に含まれる関数の役割を以下に示す。

sendSDP(), sendICE()　3.1 節で述べた方法で SDP 情報、および、ICE 情報を送信する関数。メッセージの送信には、分散型 Web ブラウザの機能が使われる。

onSDP(), onICE()　3.1 節で述べた方法で SDP 情報、および、ICE 情報を受信した時に呼ばれるイベントハンドラ。メッセージの受信には、分散型 Web ブラウザの機能が使われる。

PeerConnection()　3.2 節で述べた PeerConnection を作る関数。

addStream()　3.2 節で述べた PeerConnection に MediaStream を追加する作る関数。

onRStrmAdded()　遠隔のノードで addStream() により MediaStream が追加された時に呼び出されるイベントハンドラ。

play()　MediaStream を再生する関数。

親ノードと中継ノードは、PeerConnection()、sendSDP()、sendICE()、onSDP()、および、onICE() で、WebRTC のマルチメディアセッションを確立する。セッションが確立されると、親ノードは、addStream() により MediaStream を追加する。(親ノードがルートノードの場合、この MediaStream は、カメラで撮影した映像を含む。) 親ノードで MediaStream が追加されると、中継ノードでは onRStrmAdded() が実行される。この関数の中で、MediaStram を受け取り、変数 remote に保存し、`<video>` タグの src に MediaStream を指定し、play() で再生している。

中継ノードに新たな子ノードが加わると、親ノードと中継ノードの間でなされた手続きと同じ手続きで、WebRTC のマルチメディアセッションを確立する。セッションが確立されると、中継ノードは、addStream() により 変数

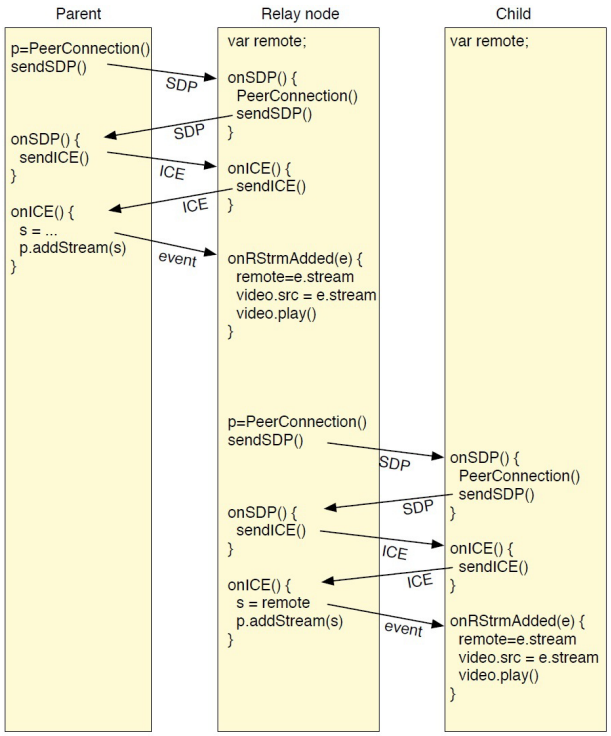

図 5　WebRTC を使ったビデオ配信の実装の概略

remote に保持していた MediaStream を追加する。この MediaStream は、親ノードから受け取ったものである。中継ノードで MediaStream が追加されると、子ノードでは onRStrmAdded() が実行される。以後、中継ノードと同様の処理が行われ、子ノード画面にビデオが再生される。

5.　ブラウザ間通信

3 章で述べたように、WebRTC の Offer と Answer は SDP 情報と ICE 情報を交換しなければならない。本研究室で開発している分散型 Web ブラウザが提供する通信機能には 2 章で述べたように SkypeRPC、FriendSocket および SocialSocket がある。しかし、SkypeRPC は、利用していた AP2AP のサービスが終了したので現在動作しない。FriendSocket が動作する Web ブラウザは Firefox 9.0 に限定され、WebRTC が動作する Web ブラウザで動作しない。まだ、ScocialSocket は未完成であり、利用することができない。

本研究ではそれらの通信手段の代替するものとして AppletSocket と AppletXmpp を実装し、利用する。

5.1　AppletSocket

AppletSocket は Java Applet の機能を利用して Java 言語の Socket クラス、および、ServerSocket クラスの機能を JavaScript 言語から利用可能にしたものである。AppletSocket を利用すると、中央サーバを介さずに、ソーシャル VPN[6][7] 通信できる Web ブラウザ間で直接通信することができる。AppletSocket は、BSD Socket と類似のインタ

フェースを提供する。BSD Socket と異なる点はメッセージの受信をイベント駆動で行うことである。

JavaScript 言語から利用可能な AppletSocket の API を以下に示す。

- creatsocket(ipaddress,port) サーバ側でソケットを生成する。
- accept() サーバ側で、クライアントを待ち受ける。
- connect(ipaddress,port) クライアント側でサーバに接続する。
- send(info) メッセージを相手に送信する。
- onrecv(message) メッセージを受信する。

上記 API を利用すると、Offer と Answer は、互いの SDP 情報と ICE 情報を中央サーバを介さずに交換できる。

5.2 AppletXmpp

AppletSocket を利用するには通信相手の IP アドレスおよびポート番号が必要である。インターネットで通信する場合はソーシャル VPN が必要になる。

本研究は、ソーシャル VPN が利用できない場合でも、インターネットで遠隔会議システムを実行するために、 AppletXmpp を実装した。これは、2 章で述べた FriendSocke と同様に、XMPP サーバを介してブラウザ間通信を行うものである。AppletXmpp は Java Applet であり、Java の Smack ライブラリ [14] を用いて XMPP サーバにアクセスし、JavaScript 用の API を提供する。XMPP サーバはユーザの認証機能を提供し、ユーザとユーザのつながりを管理する。

JavaScript 言語から利用可能な AppletXmpp の API を以下に示す。

- ConnectService(account, password) XMPP サーバにアクセスする。
- SendInfo(friend, info) 送信したい友達を指定し、メッセージを送信する。
- onRecvFriendlist(friendlist) 友達リストを受信する。
- onRecv(message) メッセージを受信する。

上記の API を利用すると、Offer と Answer 間は互いに SDP 情報と ICE 情報を XMPP サーバを介し交換できる。

6. 実験

現在、遠隔会議システムは、4 章で述べた機能のうち、単一のカメラのビデオを他のメンバに配信する機能、ファイル共有機能、および、テキスト・チャット機能が動作している。ビデオ配信機能とファイル共有機能の性能を調べる実験を行った。

6.1 実現環境

実験で用いた Web ブラウザの木構造を、図 6 に示す。実験では、2 ノードから最大 16 ノードまでノード数を変えて性能を測定した。これらの Web ブラウザのノードを、次のように 3 台のコンピュータに割り当てて実行した。

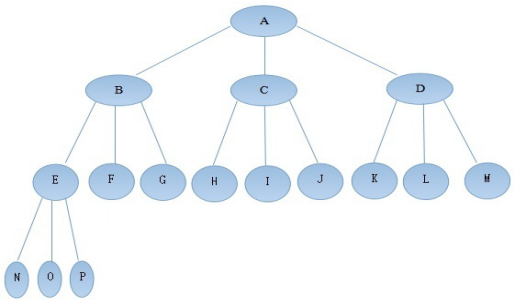

図 6 実験で用いた Web ブラウザの木構造

- コンピュータ 1: ノード A, E から M
- コンピュータ 2: ノード B, C, D
- コンピュータ 3: ノード N, O, P

このようにノードを割り当てることで、ノード間の通信は全て物理的なネットワークを経由するようにした。

本アプリケーションの性能を、LAN で測定した。分散型 Web ブラウザを実行したコンピュータのハードウェアとソフトウェアを以下に示す。

コンピュータ 1:

- ホスト OS：Windows 7 64 ビット
- プロセッサ：Core i7-920 2.67GHz
- メモリ：8.00GB

コンピュータ 2:

- ホスト OS：Windows 7 32 ビット
- プロセッサ：Core i5-750 2.67GHz
- メモリ：4.00GB

コンピュータ 3:

- ホスト OS：Windows 7 32 ビット
- プロセッサ：Core i5-M540 2.53GHz
- メモリ：4.00GB

実験に用いたネットワークを以下に示す。

- LAN: イーサネット。1Gbps。
- 分散型 Web ブラウザの通信機能: 5.1 節で述べた AppletSocket。

分散型 Web ブラウザとしては、次の Web ブラウザを拡張したものを用いた。

- Google Chrome 39.0.217.95m

6.2 実現内容

次の 2 つの実験を行った。

ビデオ配信 ノード A からビデオを配信し、他のノードで遅延を測定する。

ファイル転送 ノード A から、ファイルを他のノードに送信し、完了するまでの時間を測定する。

各実験を 2 ノードから 16 ノードまで、2 ノードずつ参加ノード数を増やすながら実行し、測定した。

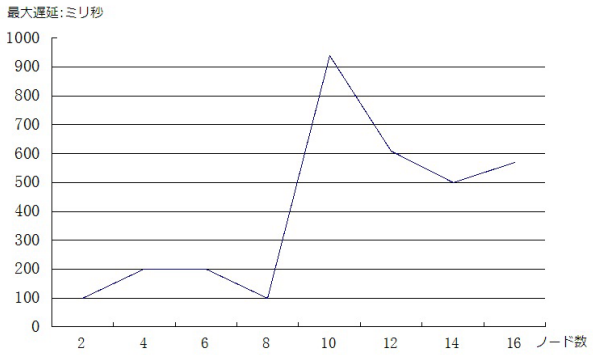

図 7　ビデオ配信における最大の遅延

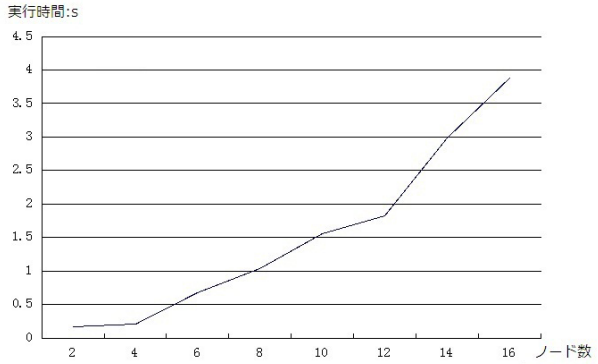

図 8　ファイル共有の最大の実行時間

6.2.1　ビデオ配信実験

この実験では A のカメラ及び最後ノードのカメラで同じ物理的な時計を撮影した。そして、A から転送されるビデオ及び最後ノードカメラのビデオを同じのブラウザで表示した。スクリーンショットを利用し、A のビデオの時計及び最後のノードの時計の静止画像を得た。2 つ時計の差を遅延とした。

実験結果を図 7 に示す。X 軸はノード数である。Y 軸は最後のノードの遅延時間である。LAN でビデオを配信した時、ノード数が 16 以下では遅延が 1 秒以内になることが分かった。ノード数が 10 の時に異常な値が出ているが、実験で利用したコンピュータの性能が不十分であったためだと思われる。今後は、より良い実験環境を構築し、さらにノード数を増やして実験したいと考えている。

6.2.2　ファイル転送実験

この実験はノード A からサイズ 418KB の画象ファイルを他のノードに配信し、実行時間を計った。実験結果を図 8 に示す。X 軸はノード数である。Y 軸はすべでノードで一番遅くファイルを取ったノードの時間である。LAN では、全てノードがファイルを受信する時間は約 4 秒であった。この実験ではノード数が 16 まではノード数が増えるに従ってほぼ線形に実行時間が増えた。WebRTC では、DTLS (Datagram Transport Layer Security) [11] と呼ばれる暗号化技術が使われており、実験では CPU の負荷が大きくなっていた。今後は、より良い実験環境を構築し、さらにノード数を増やして実験したいと考えている。

7.　関連研究

Skype は、1 対 1 のビデオによる対話に加えてグループビデオ通話を提供している。この機能を利用すれば、10 名までのユーザで無料でビデオによる通話が可能になる。Skype は、アメリカ政府 NSA により監視されていることが知られている [4]。

本研究が利用している WebRTC は、UDP を使って通信しているが、DTLS と呼ばれる、TLS (Transport Layer Security) [3] と同等の方式を使っている。したかって NSA による監視が Skype よりも困難であると思われる。また、本研究では Skype よりも多くのユーザが参加できるような遠隔会議システムを実現したいと考えている。

文献 [17] は、VCStream(Video Conference Stream) モデルに基づき遠隔会議システムを構築することを提案している。VCStream モデルでは、木構造、および、グリッド構造の 2 つの構造を組み合わせて通信路を構築する。ユーザの追加や削除等の操作は、サーバにより行われる。

この研究と比べて本研究特徴は分散型 Web ブラウザの機能を使って中央サーバを利用しないでユーザの追加や削除を行えることである。また、本遠隔会議システムは、ビデオ配信だけではなく、ファイル共有とテキスト・チャットの機能を持っている。

文献 [15][2] では、SIP (Session Initiation Protocol) を使って P2P 型ネットワークを構築し、中央サーバを用いることなく音声やビデオ通信を実現することが提案されている。これに対して本研究では、WebRTC を用いて中央サーバを用いることなくビデオ配信を実現している。

8.　おわりに

この論文では分散型 Web ブラウザにおける遠隔会議システムについて述べた。ブラウザ間のビデオデータの配信には WebRTC を利用する。WebRTC でセッションを確立すると、カメラから取得したビデオデータを中央サーバを介さずに、複数の Web ブラウザに配信する。WebRTC で通信するために必要な SDP 情報と ICE 情報を交換するため、本研究では AppletSocket および AppletXmpp を実装した。現在のところ、ビデオデータを、最大 16 人までのユーザに配信できることを確認した。

今後の課題はインターネットおける性能の測定及び同時に複数のビデオを配信する機能を実装することである。

参考文献

[1] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. “Decentralized online social networks”, In Handbook of So-

cial Network Technologies and Applications, pp.349–378, Springer, 2010.
[2] Carol Davids, Alan Johnston, Kundan Singh, Henry Sinnreich, and Wilhelm Wimmreuter: “SIP APIs for voice and video communications on the web”, In Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications (IPTcomm ’11), pp.1-7, 2011.
[3] T. Dierks and E. Rescorla: “The Transport Layer Security (TLS) Protocol Version 1.2”, RFC 5246, 2008.
[4] G. Greenwald. “No Place to Hide: Edward Snowden, the NSA, and the US Surveillance State”, Metropolitan Books, 2014.
[5] 郭飛, 青沼伴樹, 新城靖, 佐藤聡, 中井央, 板野肯三. “分散型 Web ブラウザの通信機能と協調動画視聴アプリケーション”, 情報処理学会研究会報告, システムソフトウェアとオペレーティング・システム研究会 (OS), 2012-OS-123(3), pp.1–8, 2012.
[6] 海沼直紀, 新城靖, 登大遊, 肖焜瑶, 佐藤聡, 中井央 “プライバシを保護するための VPN を用いたソーシャルアプリケーション実行環境”, 情報処理学会研究会報告, コンピュータシステム・シンポジウム（ComSys), pp.3–15, 2014.
[7] P. S. Juste, D. Woinsky, P. O. Boykin, M. J. Covington, and R. Figueiredo: “SocialVPN: Enabling wide-area collaboration with integrated social and overlay networks”, Computer Networks, Vol.54, No.12, pp.1926–1938, 2010.
[8] 水野佑樹, 新城靖, 佐藤聡, 中井央. “SocialSocket の提案”, 情報処理学会第 25 回コンピュータシステム・シンポジウムポスターセッション, 2013.
[9] 登 大遊, 新城 靖, 佐藤 聡: “SoftEther VPN Server: マルチプロトコル対応のクロスプラットフォームなオープンソース VPN サーバ”, ソフトウェア科学会 コンピュータソフトウェア, Vol.32, No.4, pp.3–30 (2015 年 12 月).
[10] 潘進睿, 新城靖, 李咏, 佐藤聡, 中井央 “分散型 Web ブラウザにおけるコメント共有アプリケーションの実装”, 情報処理学会研究会報告, システムソフトウェアとオペレーティング・システム研究会 (OS), 2014-OS-131(8), pp.1–6, 2014.
[11] E. Rescorla and N. Modadugu: “Datagram Transport Layer Security”, RFC 4347, 2006.
[12] Yasushi Shinjo, Fei Guo, Naoya Kaneko, Takejiro Matsuyama, Tatsuya Taniuchi, and Akira Sato.“A distributed web browser as a platform for running collaborative applications”, In Collaborative Computing: Networking, Applications and Worksharing, pp.278–286, 2011.
[13] L. Schwittmann, M. Wander, C. Boelmann, and T. Weis: “Privacy preservation in decentralized online social networks”, IEEE Internet Computing, Vol.18, No.2, pp.16-23, 2014.
[14] Smack. https://www.igniterealtime.org/projects/smack/, accessed: 2015-07-15.
[15] Kundan Singh and Henning Schulzrinne: “Peer-to-peer internet telephony using SIP”, In Proceedings of the international workshop on Network and operating systems support for digital audio and video (NOSSDAV ’05), pp.63–68, 2005.
[16] W3C. WebRTC http://www.w3.org/TR/webrtc/, 2013-9-10.
[17] Zhi Wang, Jizhong Zhao, Wei Xi, Zhiping Jiang. “A Scalable P2P Video Conferencing System Based on VC-Stream Model”, IEEE/ACIS 11th International Conference on Computer and Information Science, pp.77–82, 2012.