「マルチメディア，分散，協調とモバイル
(DICOMO2021)シンポジウム」 令和3年6月-7月

WebRTCによる宅内環境間P2Pラウンドトリップタイム計測

中川紘輔[1,a)]　塚田学[1]　落合秀也[1]　江崎浩[1]

概要：近年，オンラインゲームやリモート合奏などの用途でインターネット回線による通信を用いる機会が増大している．そのような通信の課題の一つとして挙げられるのが，高い同期性が必要な作業（音楽セッション等）は，従来主流であったサーバ・クライアント方式ではリアルタイム性で性能要件を満たすことができないという問題である．これに対処するために，現在高度なリアルタイム性が必要なシステムでは，エンドツーエンドを直接つないで遅延を抑制できるP2P方式が採用される場合がある．しかしながら宅内ネットワーク環境間P2P通信性能については先行研究が少なく，遅延やビットレートといった通信性能を多様な宅内ネットワーク環境下で計測することは，このようなシステムの普及推進において重要である．これを明らかとするために，本研究では，WebRTCの統計情報取得APIを用いたP2P性能計測ツールを開発し，宅内ネットワーク環境間のP2P性能を，クライアントアプリケーションのダウンロード等を行うことなく，Webブラウザベースで簡易に計測する手法を提案する．また，この計測を多ノードフルメッシュ接続にも対応させ，多ノードフルメッシュ接続の性能限界について統計情報から考察を行う．

The measurement of P2P Round-Trip-Time between on-home networks

Kosuke Nakagawa[1,a)]　Manabu Tsukada[1]　Hideya Ochiai[1]　Hiroshi Esaki[1]

1. はじめに

近年，オンラインゲームやリモート合奏などの用途で，従来のようなクライアント・サーバーモデルの通信ではなく，P2Pの通信を用いたサービスを提供する事例が散見される．このような通信は，サービスを使用するクライアント間をサーバーを介さずダイレクトに接続することで，クライアント・サーバーモデルによる通信よりも遅延を減らし，リアルタイム性を高めることを企図して使用されている．例えば，Yamahaの「Syncroom」[4]は，オンラインで音楽セッションをすることを目指したソフトウェアで，低遅延の音声のリアルタイムストリーミングを提供している．一般に，音楽セッションで許容される遅延は高々10ms台のオーダーで，Zoomの遅延が100〜200ms程度であることを考えると，一般的なサーバ・クライアント型でのシステムでは達成困難な水準である．これを解決するために，Syncroomでは利用者間をP2Pでフルメッシュ接続を行い，遅延を可能な限り減らす実装を行っている．このように，P2P通信は用途によっては高度なリアルタイム性を担保するために不可欠な手法となりつつある．

しかし，SyncroomのようなP2P音声ストリーミングソフトや，高度な同期が必要なオンラインゲームを多ノード間で運用する場合，ある一つのノードの遅延が非常に大きくなってしまう事象が発生した場合に，参加者全体が不利益を被ってしまう問題が考えられ，遅延の計測や，遅延が特に大きいノードの特定といった作業はユーザビリティ向上のために非常に重要である．このような問題意識はオンラインゲームを提供する企業においても持たれており，Sharadら[11]はP2Pゲームにおいて低遅延なマッチングを実現するための，地理情報に基づいたP2P遅延予測システムを提案している．

しかしながら，P2Pの遅延を惹起する要因は地理的要因だけではなく多岐にわたるため，ユーザビリティ向上のためには事前の遅延予測だけでなく，実際に通信を行って生じる遅延を統計情報として収集・分析することも重要なア

[1] 東京大学
The University of Tokyo
[a)] nakagawa-kosuke973@g.ecc.u-tokyo.ac.jp

プローチであると言える．現状ではこのような通信の品質計測は，各サービス運用者が独自に調査を実施したり，統計情報を収集したりすることが多い．Youngki らの調査 [16] では Xbox 360 の 560 万ユーザーに対して P2P 通信品質を計測することで，かなり大きな統計情報を収集しており，一定の成果を挙げている．しかしこの手法は 13 年前の取り組みで，データが古いのに加えて，サービス運用者による取り組みであるため，Xbox 360 ユーザーという限定的な集団での計測しか行うことができていない．目下の新型コロナウイルス拡大によるリモート活動増加に伴い，今後とも P2P によるリアルタイム性の高い通信はユースケース・利用人口ともに増加していくと考えられる．したがって，より一般的・体系的に統計情報を収集・分析する方法を検討することは公益性が大きい課題であると言える．

また近年では，WebRTC というブラウザ上で P2P のメディアストリーミングをサポートする技術が登場しており，ブラウザベースでビデオチャットアプリケーションを簡単に実装できる利点から，利用が拡大している．この WebRTC は，P2P の統計情報の計測をサポートしており，上記のような研究課題とは親和性が高い．

本研究では，上記の背景から，WebRTC を用いた，ブラウザベースで宅内環境間の P2P 通信性能を計測するツールを開発した．具体的には，ブラウザから計測用 Web サイトにアクセスを行うと，JavaScript のプログラムにより WebRTC が呼び出され，WebRTC において実装されている統計情報収集 API を使用することで，宅内環境間の P2P 通信計測がブラウザベースで実行されるというものである．これにより，実際の宅内環境間を結んだ計測を，クライアントアプリケーション等をインストールすることなく，簡易的に行うことを目指した．

また，ピア同士の個別の P2P 通信性能計測ではなく，多ノードフルメッシュ接続に対応したツールを製作することにより，効率よく広範な統計情報を行うことを目指した．

また，収集したフルメッシュの通信性能データを，実際に宅内環境間 P2P 通信の問題解決に役立てるという観点から，生のデータに対して直感的な問題の把握が可能な可視化を行う手法を提案した．

2. WebRTC 概要

WebRTC とは Web Real-Time Communications の略で，ブラウザやモバイルアプリに簡潔な API を通じてリアルタイム通信を提供するオープンソースの技術である．この API は仲介を必要とせずにブラウザー間で直接，任意のデータの交換や，キャプチャしたオーディオ／ビデオストリームの送受信を可能にする．また，このストリーミングは P2P を介して行われる．WebRTC は以下の技術を用いてブラウザ間の P2P 通信を実現している．

2.1 ICE

ICE とは Interactive Connectivity Establishment の略で，ブラウザ間で P2P 通信をする際に NAT 越えの機能を担うプロトコルである．このプロトコルはオファー・アンサーモデルを使用する任意のプロトコルで利用可能である．ICE は STUN と TURN という 2 つのプロトコルを使用する．

- STUN：LAN 内のクライアントに，WAN 側から観測できる IP アドレスとポート番号を伝達するプロトコル
- TURN：NAT 越えを要する通信の間に TURN サーバーを挟んで，相手のピアとの通信を仲介させるプロトコル

これらを用いて ICE は，LAN 内のノードに対し，NAT を越えた P2P 通信を補助する手段を提供する [13].

2.2 シグナリング

WebRTC で NAT を超えた P2P 接続を行うためには，上述の ICE，およびセッション情報について記載された SDP 等の情報をピア間で共有する必要がある．これは WebRTC によって NAT を超えた P2P 通信が確立される前に，一度 NAT を超えて ICE をピアに送信する必要があることを意味し，これを実現するには WebRTC 以外に別途 NAT 越えを可能とする通信手段を用意しなくてはならない．

SDP，ICE をピア間で共有する機能をシグナリングといい，この機能は WebRTC ではサポートされておらず，別途その機構を準備する必要がある．このような機能は多くの場合，ネットワーク上に WebSocket 通信を行うサーバーを用意することによって実現される．

2.3 WebRTC の MediaStream

WebRTC の MediaStream におけるパケットは RTP を用いて送信される [8]．RTP は UDP 上で動作するアプリケーション層のプロトコルであり，用途に応じて自由に機能が付与できる拡張性が備えられていると説明されている [14]．Google Chrome 等のブラウザではこの拡張を行うことにより，WebRTC 統計情報取得などの機能を実装している．

2.4 WebRTC のデータチャネル

WebRTC のデータチャネルは現在，RFC で標準化過程にカテゴライズされている．

RFC 8831[10] に記載されている具体的な仕様は 2.4.1, 2.4.2 に示す．

2.4.1 通信方式

WebRTC のデータチャネルは「SCTP over DTLS over ICE/UDP」というカプセル化が行われて送受信される．

UDP はトランスポート層のプロトコルの一つで，動画や音声のストリームに用いられることが多い．UDP は TCP

と異なり，ハンドシェイクや再送制御を行うことがなく，信頼性や順序性などが担保されない通信方式である [7].

DTLS はトランスポート層における暗号化プロトコルであり，通信路における盗聴や改ざんを防止する役割を果たす [12].

SCTP はピアとピアを結ぶ 1 つのストリームの中で仮想的にマルチストリーミングを行うことが可能で，各ストリームに対して信頼性や順序性を担保するかしないかを選択することが可能である．TCP と UDP の両方の特性を合わせ持ったプロトコルであるといえる [15].

WebRTC のデータチャネルは後述のように順序性を担保するかしないかを選択して通信を行うことが可能であり，これは SCTP を用いることによって実現されている．

2.4.2 ユースケース

WebRTC のデータチャネルは dataChannelOption の ordered で bool 値を指定することで，高信頼性モードと低信頼性モードの二種類のいずれを用いるか選択することが可能である．

高信頼性モードでは送信データの順序性が保証され，低信頼性モードでは順序性が保証されない．高信頼性モードはゲームの状態情報やファイル，テキストチャットなどの順序性が必要なデータ通信に利用され，低信頼性モードではリアルタイム更新されるゲームの位置情報などの順序性がそこまで重要ではない通信に用いられる．

2.5 統計情報に関連する実装

WebRTC の統計情報は, RTCPeerConnection.getStats() メソッドによって取得することが可能だが，ブラウザ間で標準化がなされておらず，得られた生の統計情報に対する処理をする必要がある．

オープンソースのツールとして，以下のものがあり，本研究ではこれを使用することによって統計情報の取得を行う．

2.5.1 SkyWay

SkyWay は，株式会社 NTT コミュニケーションズが提供する WebRTC のソフトウェア開発キットであり，STUN サーバ等の P2P 通信に必要な機器を提供するとともに，それらを使用できる SDK も提供している．

2.5.2 rtcstats-wrapper

このツールは SkyWay 公式によってサポートされている WebRTC 統計情報のラッパーである．これを用いることで，

- WebRTC 統計情報のブラウザ標準化
- WebRTC 統計情報の瞬間値計測
- WebRTC 統計値の定期監視

が可能となる．[5]

3. 関連研究

WebDINO VideoMark[6] は，ブラウザ拡張機能で，YouTube などの動画配信サービスの体感品質を計測する取り組みである．ここで計測する体感品質（QoE）とは，解像度やフレームレート，バッファリング，一時停止などの客観的な情報から，利用者が体感する主観的なサービス品質を推定したものである．この拡張機能はビットレート，解像度，再生品質情報，ISP などのネットワーク情報，クライアントの端末情報等のデータを収集し，通信事業者や動画配信サービス事業者のサービス品質改善に用いられる．

WIDE で研究されている SINDAN[3] は，ネットワーク運用者がクライアントから受ける「つながらない」というクレームに対し，従来のオペレーターによる電話対応のようなコストが大きい手法や，Nagios 等のオープンソースアプリケーションに代わる障害点特定手法を開発するプロジェクトである．このプロジェクトは，ユーザーが利用している実環境からの状態観測情報をネットワーク運用者に的確に伝える手法の確立を目指している．

RIPE Atlas[1] は，世界各地に監視ノードを配置することで，世界各地のインターネット回線の遅延やトラブル情報を取得し，可視化するプロジェクトである．Ripe Atlas では，同意が得られた世界中の参加者に対して監視ノードを無償で送付し，それを LAN 内に設置してもらうことで，LAN 環境からの Pingping，traceroute，SSL / TLS，DNS，NTP，および HTTP 等の計測情報を自律的に収集し，世界中のネットワーク環境の接続性や到達可能性のモニタリングを目指している．

iNonius Project は組織内部専用で使用できるインターネット計測サイトを開発することで，従来よりも正確で迅速なインターネット計測環境を整備することを目指している [2].

Shima らの Home Area Measurements[17] は，宅内ネットワーク計測を Web 上で簡単に行うために，(ブラウザから計測サーバーまでの遅延) – (計測サーバーから最遠の tracepath) を計測することで，宅内ネットワークの遅延を推定する手法を用いたサービスを提供することを目指している．

Romain らの Persistent Last-mile Congestion: Not so Uncommon[9] では，traceroute のデータを活用してラストワンマイルの混雑状況，および通信のボトルネックを明らかにすることを目指している．

先行研究と本研究で作成するツールの比較は表 1 に示す．

4. 提案手法

本研究では上記のオープンソースのツールを元に，以下

表 1 先行研究と本研究の比較

研究	調査対象	P2P 計測	計測環境の制約	計測数
WebDINO VideoMark[6]	動画ストリーミングの体感品質	サーバ・クライアント	-	-
Sharad らの手法 [11]	ゲームのリアルタイム通信性能予測	予測手法の提案	-	-
Youngki らの調査 [16]	ゲームのリアルタイム通信性能	✓	Xbox360 に限定	560 万
本研究	ゲームや音声等のリアルタイム通信性能	✓	Web ブラウザ	同時接続 10-30 人

のような要件を満たしたツールを製作する.

4.1 機能要件

4.1.1 クライアントアプリケーションをインストール不要なツール

従来，P2P 通信というのは専用のクライアントアプリケーションをインストールする形態で利用されることが多い．しかし，本研究の目的は性能計測であり，専用のクライアントアプリケーションを用いて計測するのはユーザーに多大な労力をかけてしまう上に，広範な環境下での情報収集は困難である.

よって本研究では，この計測のために各ユーザーが専用のクライアントアプリケーションをインストールしなくてもよい計測手法を提案する．具体的には，多くのコンシューマー向け PC にデフォルトでインストールされている Web ブラウザ上で計測を完結させることにより，簡便な計測を実現した.

4.1.2 軽量に動作する計測ツール

本手法で計測したい統計情報は，最終的には動画や音声などの安定的なストリーミング等を目指した問題解決に役立てられることを想定しているが，計測段階で実際に動画や音声のストリームを行いながら計測することは困難である.

理由としては，動画や音声のストリームを多ノード間で行った場合，回線への負荷や CPU・GPU 負荷が非常に大きくなってしまい，正常な計測を行うためには参加人数が大きく制限されてしまうことが挙げられる.

このような事情により，本研究では WebRTC の MediaStream ではなく，DataChannel を通じて少量のデータを一定間隔で送信し続ける仕様とした．これにより，上述のように各ノードや回線に過剰な負荷がかかることなく，比較的軽量かつ安定した計測を実現した.

4.1.3 多ノードをフルメッシュで接続，計測

1 対 1 のペアで P2P 通信性能を計測する手法では網羅的な統計情報収集には大変な労力を有する．同時に多ノード間の通信性能を計測することができれば，統計情報収集が可能なピアの数はノード数の二乗に比例して増加する．これによりデータ収集の大幅な効率化を行うことが可能であるため，多ブラウザ間をフルメッシュ接続するような仕様のシステムを製作した.

4.1.4 パケットサイズを変えた計測

WebRTC の MediaStream による動画・音声のストリーミングの負荷は固定である．動画・音声のストリームによって統計情報の収集を行うことは，本来の用途という観点では目的に見合った計測手法であるが，CPU や GPU 負荷の制約，帯域の問題等により多ノードが接続する計測ツールとは相性が悪い.

本研究では MediaStream の代わりに DataChannel を用いて複数のサイズのデータを送信することで，ハード面や回線の負荷を抑制しながら多ノード同時計測を行い，なおかつ負荷を可変とした計測を行い，通信の負荷と品質の特性を知ることができるようなシステムを製作した.

4.1.5 パケット送信頻度を変えた計測

上記同様，負荷を可変とするためのもう一つのパラメータとして，パケット送信頻度も可変とすることで，2 つのパラメータの作用の差異についての考察を試みた.

4.1.6 可視化

多ノードフルメッシュ接続の統計情報は，生データのままでは非常に複雑であり，本研究の計測を P2P 通信性能面での問題解決に役立てるという本来の趣旨に結び付けるためには，何かしらのデータ可視化手法が不可欠である.

今回はヒストグラム等の古典的な可視化手法のみならず，メッシュ接続の図をベースとし，P2P 接続情報を容易に把握することが可能な可視化を行うことを目指した.

4.2 システムの概略

上記の点を踏まえて，本研究で製作するシステムの概略を図 1 に示す.

本手法の全体像としては，1 に示すように,

(1) WebRTC によるブラウザ上での P2P 性能計測
(2) 計測データをサーバーで収集
(3) 可視化

の三段階に大別される.

ピアごとの計測については,「NAT を越えた P2P 接続において性能計測をする」ということが，WebRTC を用いることでブラウザベースで実現される仕様とした.

多ノードのフルメッシュ接続については，Web 上の異なる位置に存在する複数の宅内環境をフルメッシュで接続・計測を実施することを目指した.

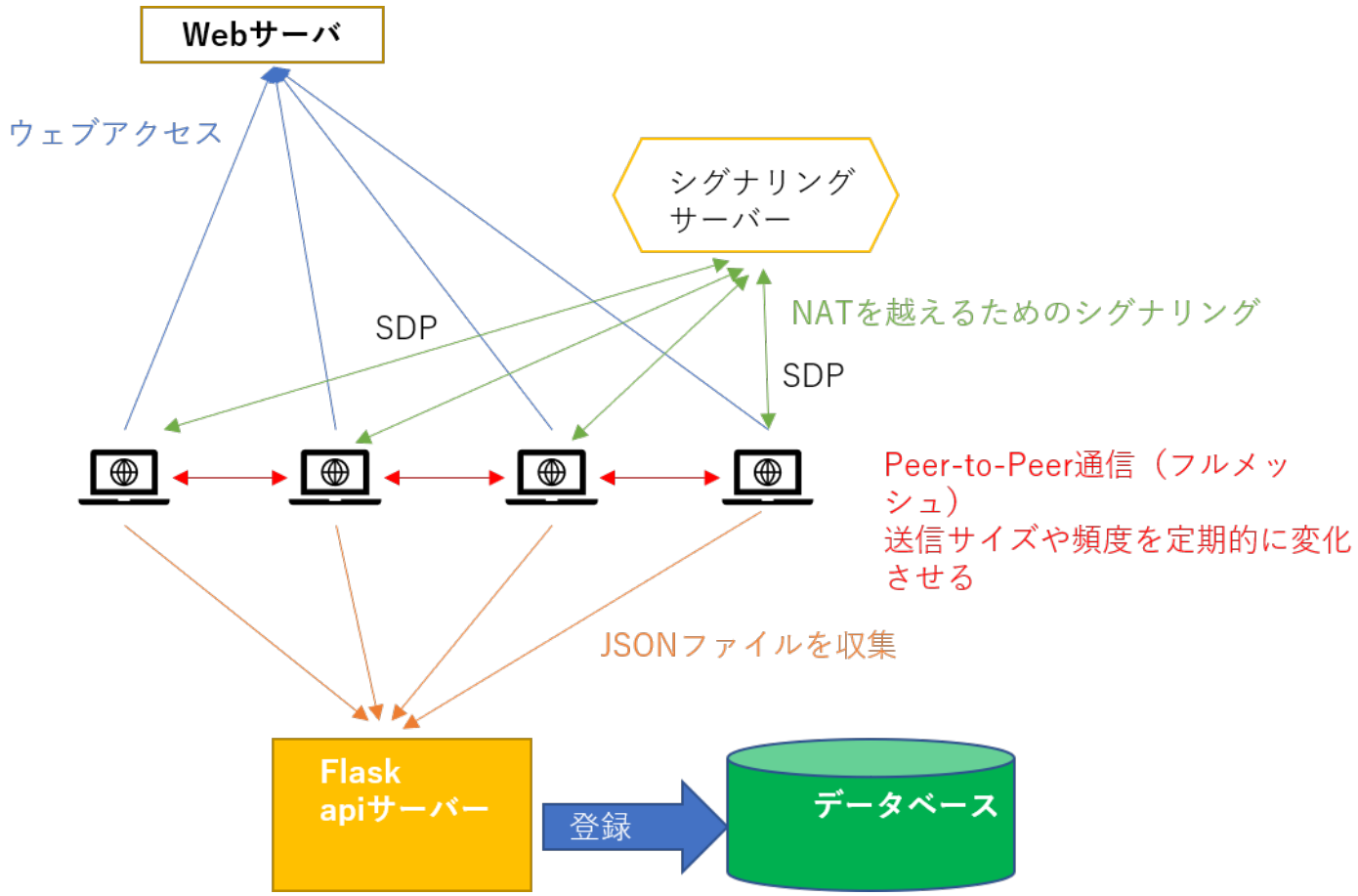

図 1　システムの概略

4.3　WebRTC による P2P 性能計測ツール

計測ツールの具体的な実装について紹介する．

この計測ツールでは，

(1) シグナリングサーバー

(2) WebRTC による P2P 通信を行う Web フロントエンド

の 2 種類の機能を実装した．

4.3.1　シグナリングサーバー

シグナリングサーバーは，Node.js を用いて WebSocket でクライアント間の SDP,ICE をやり取りする実装とした．シグナリングサーバーはクライアントの JavaScript プログラムとの間で以下のような SDP,ICE の交換を行う．

(1) 新たなクライアントが接続した場合は，WebSocket 側で id を自動生成し，送信された room 名ごとに管理する．

(2) クライアント側から SDP が message として送信されてきた場合は，room ごとに管理されている接続中の id すべてに SDP を broadcast する．

(3) 自分がルームに入室した場合はルーム全体に ICE を broadcast し，入室済みノードが新規に入室したノードから ICE を受け取った場合は，自分も ICE を送り返す（Trickle ICE）

(4) クライアント側から切断が通知された場合は，room ごとに管理されている接続中の id すべてに切断の通知を broadcast する．

シグナリングサーバーは，このようにあるクライアントから送られた SDP,ICE や接続・切断等のイベントを，room 内の他クライアントに broadcast するような機能を果たすことで，クライアント同士の P2P 接続に必要な情報を伝達する．

4.3.2　WebRTC による P2P 通信を行う Web フロントエンド

計測ツール（Web サイト）を開いた際の画面は図 2 のようなものとした．WebRTC の通信として，動画や音声のストリームを行う MediaStream と，ファイルやデータのやり取りを行う DataChannel が存在するが，本手法では，DataChannel を用いて軽量な通信を行うことで，多人数による同時計測が可能な実装を目指した．

なお，本実装では DataChannel の通信モードは順序性を担保する高信頼性モードとした．

また，ブラウザ機能から得ることができる RTT 値だけではなく，https://ipinfo.io にリクエストを送って返された各ノードのグローバル側から観測される IP アドレスと，アンケート調査に基づいた各ノードのデータリンク層の接続情報（有線接続か無線接続か）も取得を行えるように

図 2　計測ツールの Web フロント

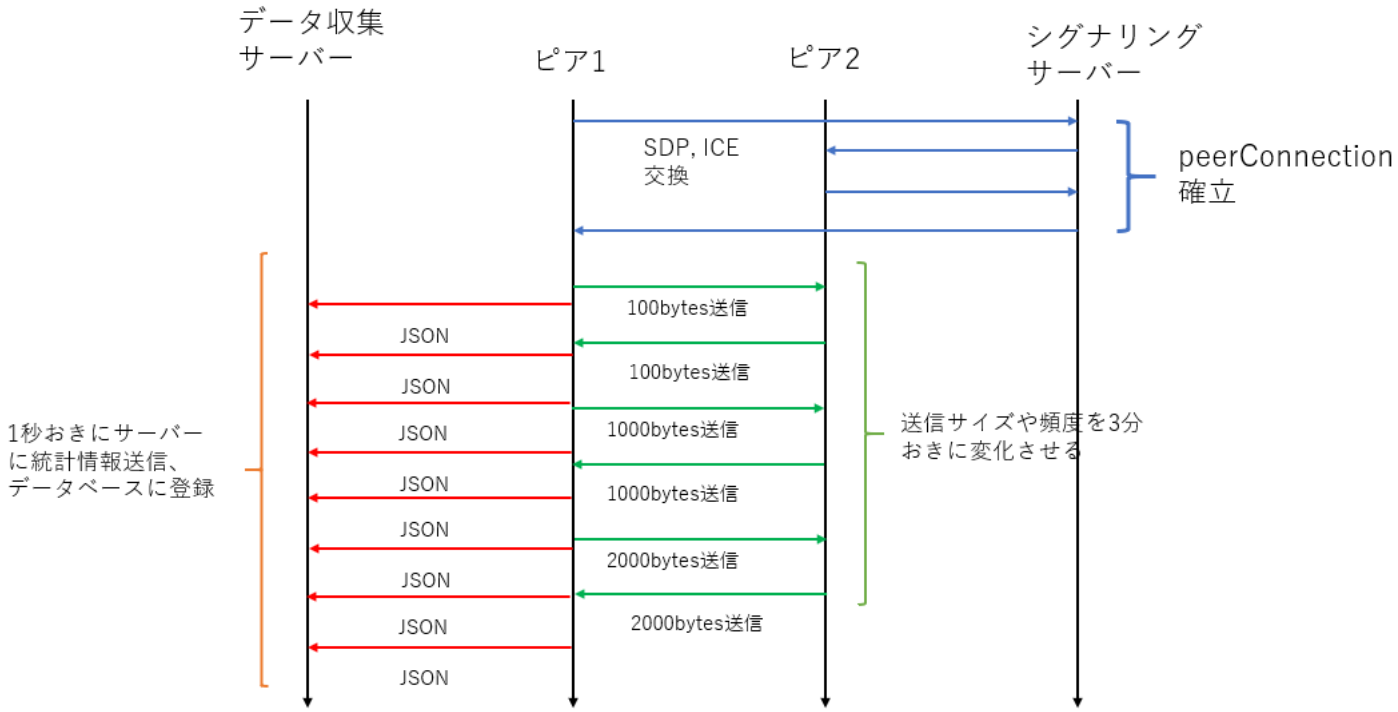

図 3　システム全体の時系列

した．

このように各クライアントのブラウザ上で取得されたデータは，1 秒おきに後述のデータ収集サーバーに POST される．

計測ツール全体としては，図 3 に示す時系列で動作する．

4.4　データ収集サーバー

各 Web フロントで得られた計測データの収集を行うサーバーとして，Python の軽量フレームワークである Flask を使用し，図 3 に示すような api サーバーを作成した．

各クライアント側から 1 秒おきに POST された統計情報は，サーバー側でデータベースに登録し永続化される．これにより，各ノードの情報を一元的に解析することを可

能とした．データ収集サーバーで収集される情報は図 3 に示す．

表 2 収集される情報

id	統計情報固有の id. サーバー側で自動割当
roomId	フルメッシュで繋ぐ room 名.
yourId	各クライアントの識別子
peerId	P2P 接続相手の yourId
yourId_peerId	yourId + '_' + peerID
date	Date() で取得された日付 [msec]
yourIp	ノードのグローバル IP アドレス
yourIsp	ノードが接続している回線のプロバイダ
candidatePair_rtt	P2P 接続のラウンドトリップタイム [sec]
datalink	端末の接続方法
currentSendBytes	送信サイズの大きさ [bytes]
currentSendTime	送信間隔 [ms]

4.5 可視化ツールの実装

上記のデータ収集サーバーで一元的に収集されたフルメッシュの P2P ラウンドトリップタイムの統計情報は，生データのままでは非常に複雑なままで，単純な有向グラフ等の可視化を行っても非常に見づらいという問題がある．

よって，本研究では人間がそれを見て P2P 性能の傾向を把握し，問題解決に役立てることができるような可視化手法を提案する．

本研究が想定する問題解決の手法として，以下のものが挙げられる．

- P2P 性能が悪いピアの組み合わせの特定：P2P ラウンドトリップタイムはピア同士の物理的距離やネットワーク的距離に依存することが考えられ，多ノード同時接続時に相対的に「相性が悪い」ピアの存在が考えられる．相性が悪いピアの存在は，ルーム全体の音声ストリーミングの品質等に悪影響を及ぼすことが考えられ，それを可視化することにより，該当ピア同士の接続を切断する，等の対処を講じることができる．
- 各ノードごとの通信品質の傾向の把握：P2P ラウンドトリップタイムは各ノードの LAN 環境にも依存することが考えられ，そのような場合は他の複数ノードとの通信品質に一定の傾向か見られる可能性がある．「他ノードに対して，全体的に通信品質が悪い」ノードを発見することができたら，LAN 環境の見直し等の対処を講じることができる．

そこで，得られたフルメッシュの P2P ラウンドトリップタイム統計情報に対し，ヒートマップによる可視化を行う．ヒートマップの可視化では，P2P 通信として実用に耐えうる性能区間として，0-100[ms] と範囲を絞ったうえで着色を行い，100[ms] を超えるものや，接続が行われなかったノードは白色とした．

5. 実験

5.1 実行環境

表 3 実行環境

Web サーバー	Ubuntu 18.04/Apache2
シグナリングサーバー	Ubuntu 18.04/Nodejs
サーバーの所在地	東京大学工学部二号館 10 階
Web ブラウザ	Google Chrome

以下，二度にわたって行われた実験の結果を紹介する．なお，この二度の実験ではそれぞれ異なった条件で計測を行っており，一回目は送信サイズを固定して送信間隔を 3 通り変化させて計測，二回目は送信サイズを 3 通り変化させ送信間隔は固定して計測を行っている．

実験に使用した環境は表 3 に示す．

5.1.1 実験 1（送信間隔 [ms] を変化）

表 4 の条件で計測を行った．図 4, 図 5, 図 6 を見ると，RTT の最頻値は 0-100[ms] の区間内であり，大半の RTT は 2 桁 [ms] であるとわかる．

また，およそ 400[ms] より大きい RTT 値が，各区間ごとに 1-10 個のオーダーで存在している．これらは全体に対する割合的には通信品質に大きな影響を与えることはないと考えられる．

また，図 10, 図 13 のみ接続ノード数が大きい（14 というノードが存在している）事象に関しては，測定開始時には実際にこの数のノードが接続していたが，時間変化とともに接続数が減少したことに起因すると考えられる．この理由としては人為的操作，あるいは Google Chrome がタブを自動リロードすることにより接続が切断されてしまった可能性が高いと考えられる（本研究の計測手法の不安定性は 7.1.2 を参照）

図 7 によると，RTT の時間変化については，接続を開始した瞬間にはばらつきが大きくなっているものの，定常的には大きな変動を確認することはできなかった．

図 8 と図 9，図 11 と図 12 を比較すると，全体的な RTT 値の傾向に大きな変化はないことがわかる．

送信間隔 1000[ms] の図は，相対的に低遅延のノードが多いことから，送信間隔がより長いほうが RTT 値が改善する傾向があると考えられる．

5.1.2 実験 2（送信サイズ [bytes] を変化）

表 5 の条件で計測を行った．図 14, 図 15, 図 16 を見ると，RTT の最頻値は 0-100[ms] の区間内であり，大半の RTT は 2 桁 [ms] であるとわかる．

しかし，およそ 100[ms]-300[ms] 程度の RTT 値が，各区間ごとに 100 個程度のオーダーで存在している．これは通信品質の悪いエッジが長い時間接続を継続したことにより，コンスタントに大きな RTT 値を記録し続けたことによるものと考えられる．

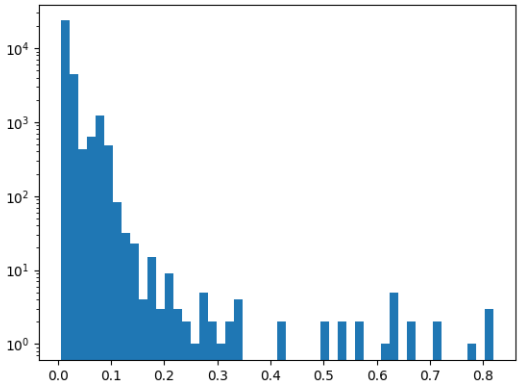

図 4 100byte-100msec の全取得 RTT のヒストグラム

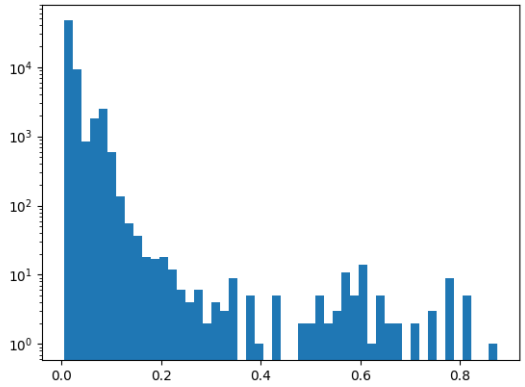

図 5 100byte-300msec の全取得 RTT のヒストグラム

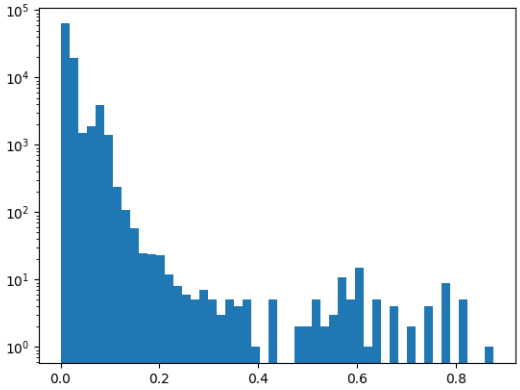

図 6 100byte-1000msec の全取得 RTT のヒストグラム

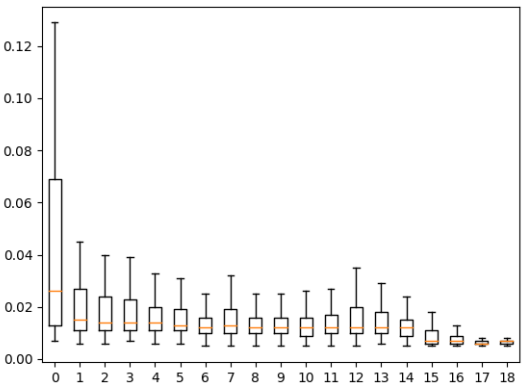

図 7 100byte-1000msec の RTT の時間変化（9 分毎）

表 4 実験 1

実験日時	2021/1/26 18:00-21:00
実験参加者	江崎落合研究室の教員，学生（最大同時接続 14 名）
送信サイズ	100[bytes]
送信間隔	100,300,1000[msec]
Web ブラウザ	Google Chrome

表 5 実験 2

実験日時	2021/1/27 19:00-21:00
実験参加者	江崎落合研究室の教員，学生（最大同時接続 10 名）
送信サイズ	100,1000,2000[bytes]
送信間隔	1000[msec]
Web ブラウザ	Google Chrome

送信サイズ 100[bytes] の図 18，図 21 は，相対的に青や緑のノードが多いことから，送信サイズがより小さいほうが RTT 値が改善する傾向があると考えられる．

Ethernet フレームの最大長が約 1500[bytes] であり，そのサイズを超えることによる影響については，色別のクラス分けで顕著に差がみられるということはなく，本研究の計測条件や可視化手法では差異を検出することは困難であった．

図 17 によると，RTT の時間変化については，定常的には大きな変動を確認することはできなかった．

可視化した結果としては，図 18-図 23 において橙・赤色で表示されている 2 番のノードは通信品質が極めて悪く，P2P 通信を使用したメディアストリーミングを行うことが困難であると考えられる．可視化した結果としては，図 18-図 23 において橙・赤色で表示されている 2 番のノードは通信品質が極めて悪く，P2P 通信を使用したメディアストリーミングを行うことが困難であると考えられる．

図 18 と図 20，図 21 と図 23 を比較すると，色が変化しているノードが目立つことから，全体的に送信バイト数を増やすと RTT が悪化する傾向にあると考えられる．

6. 考察

6.1 Web ブラウザ上で計測を行うことの長所について

WebRTC を活用した計測は，多くのパソコンにデフォルトでインストールされている Web ブラウザ上で計測のすべてを完結させることが可能であり，上記のような問題を全て乗り越えて計測を行うことが可能であり，この点においては有効性が高いといえる．

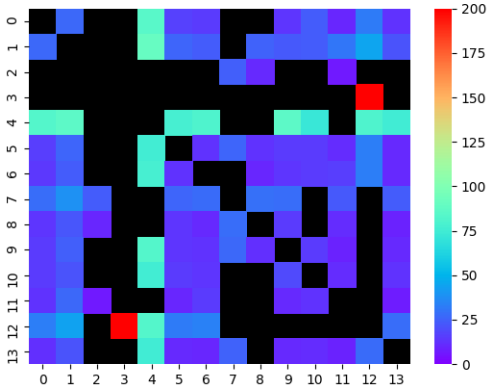

図 8 100byte-100msec のピア毎の RTT 平均値

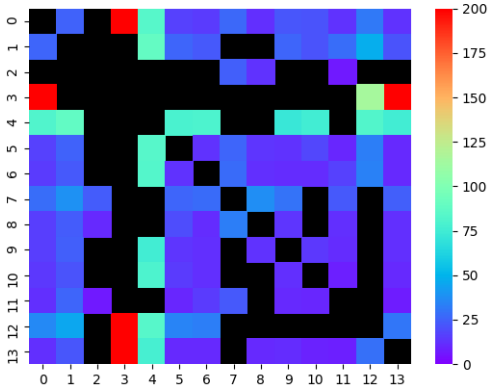

図 9 100byte-300msec のピア毎の RTT 平均値

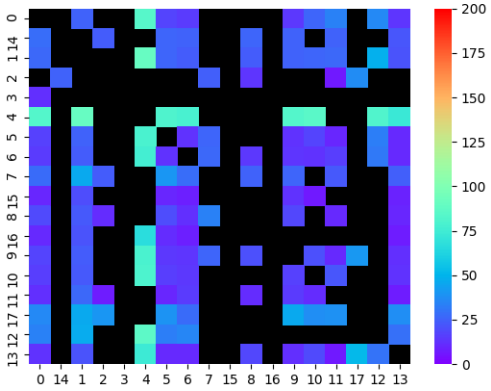

図 10 100byte-1000msec のピア毎の RTT 平均値

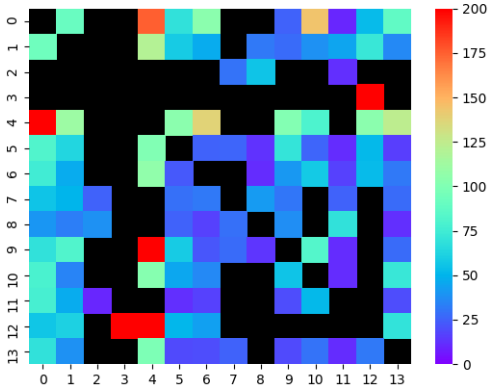

図 11 100byte-100msec のピア毎の RTT の 99%点

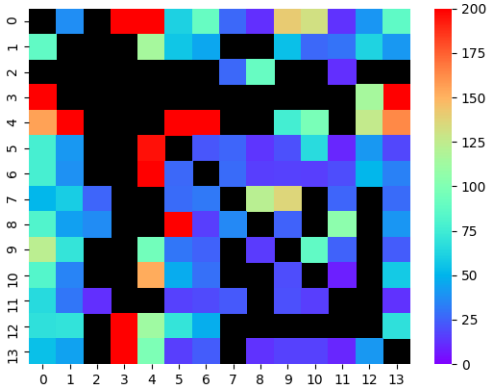

図 12 100byte-300msec のピア毎の RTT の 99%点

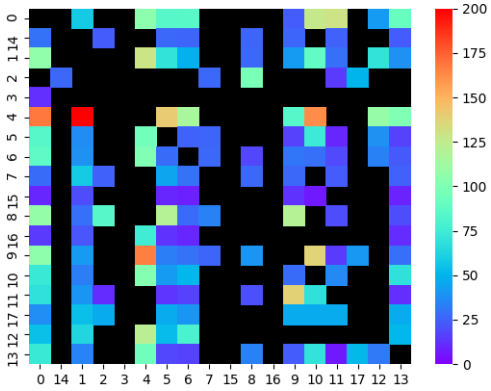

図 13 100byte-1000msec のピア毎の RTT の 99%点

6.2 データチャネル通信性能計測ツールにおけるデータ送信頻度・サイズという二種のパラメータについて

宅内環境間 P2P 通信は，動画や音声のストリーミングを低遅延で行うことを目的として利用されることが多く，このような用途では UDP パケットを連続で送信し続けるような通信が想定される．

WebRTC のデータチャネル通信は send 関数が実行されたタイミングで一つのまとまったデータが送信される仕様であり，ストリーミングのパケット送出とは異なった性質のパターンを持つ可能性が高い．

7. 今後の展望

7.0.1 宅内環境間 P2P 性能のよりリアルタイムな可視化

今回は取得した RTT をデータベースに蓄積した後，SQL 文を実行してデータを切り出してグラフを作成するという可視化を行ったが，Web アプリケーション内で完結してい

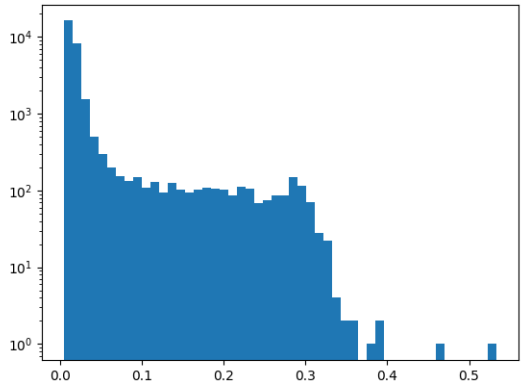

図 14 100byte-1000msec の全取得 RTT のヒストグラム

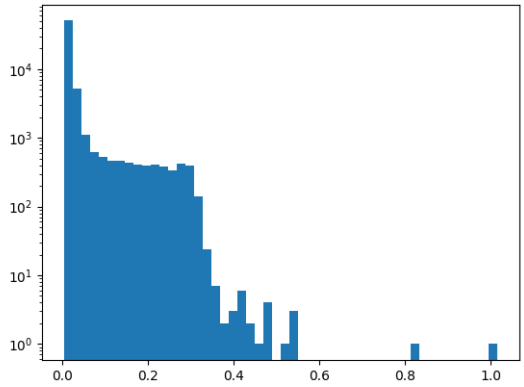

図 15 1000byte-1000msec の全取得 RTT のヒストグラム

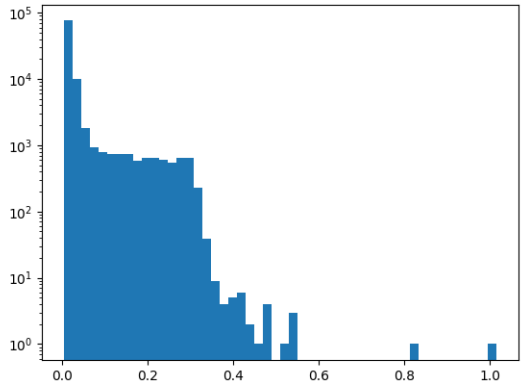

図 16 2000byte-1000msec の全取得 RTT のヒストグラム

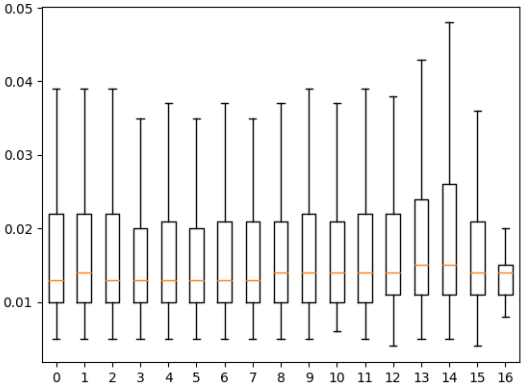

図 17 100byte-1000msec の RTT の時間変化（9 分毎）

ない以上，多くの一般ユーザーが気が向いたときに自由に計測を行い，P2P 性能を知る手段を提供できているとは言えない．

宅内のネットワーク性能を調査するツールとしては，Netflix の fast.com 等が挙げられるが，このようなツールは使用しているユーザーがリアルタイムに宅内ネットワークの状況を知ることが可能であるという点において，企業とユーザー双方に利益がある．

本研究の計測は Web 会議などの機会に，多くの参加者に協力を呼び掛けることによってはじめて実現したが，Web アプリケーション内で可視化プロセスを完結させ，ユーザーに直接 P2P 通信性能を通知する機能を提供することができれば，より広範な利用へとつながり，同時に広範なデータを収集することが可能であると考えられる．

7.0.2 計測時間や内容の見直し

本研究の計測ツールは Web ブラウザ上で動作することから，計測時に予期せぬ挙動を示すことが多かった．具体的には，ブラウザの機能でツールが処理落ちしたと判定されてタブが自動リロードされたり，ピア間の接続条件が極めて悪い時には予期せぬ形で P2P 通信が切断されてしまうことがあった．

このような事象を回避し，安定的な動作を実現するには実装レベルでの例外処理等を充実するしかないと考えられるが，ブラウザの実装もまた発展途上であるので，本研究のような長時間計測を安定的に行うのは限界があると考えられる．

fast.com のようなスピードテストのように，短時間で単発的に実施したり，送信パケット数を少なくしたうえで，より多くの人が多くの回数施行することによってデータを蓄積する実装にする必要性があるものと考えられる．

7.0.3 幅広い時間帯での計測

本研究では時間変化による RTT 値の変動を検出することも目指したが，結果的には高々三時間程度のスパンでは顕著な変動を検出することができなかった．

より広範な時刻，および平日や土日等の条件において統計情報を蓄積し，時間変化による変動について検討する必要があるものと考えられる．

7.0.4 地理的距離が大きい条件での計測

宅内環境間 P2P 性能は地理的距離にも大きく依存する．Web ブラウザ上で動作する本研究のツールは，遠隔地や海外においても計測の協力を仰ぐことが（クライアントアプリケーションによる計測よりも）比較的容易である．国境を越えた P2P 性能に関する研究は先行文献も極めて乏し

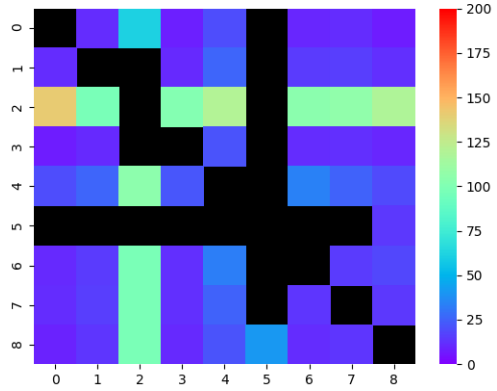

図 18 100byte-1000msec のピア毎の RTT 平均値

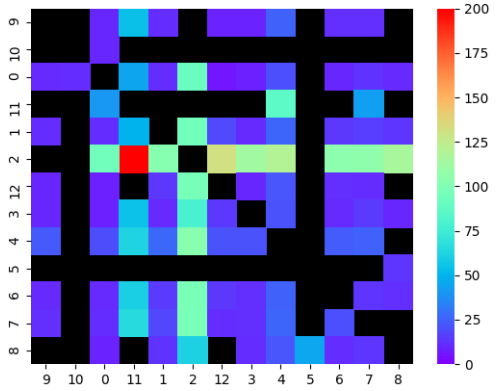

図 19 1000byte-1000msec のピア毎の RTT 平均値

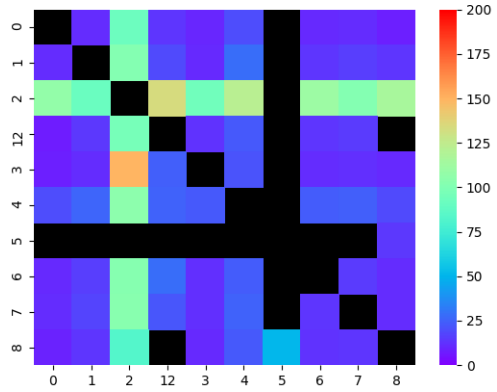

図 20 2000byte-1000msec のピア毎の RTT 平均値

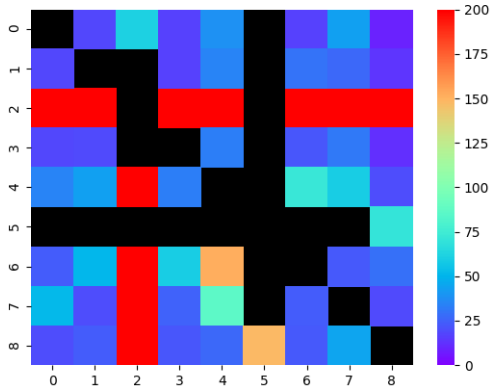

図 21 100byte-1000msec のピア毎の RTT の 99%点

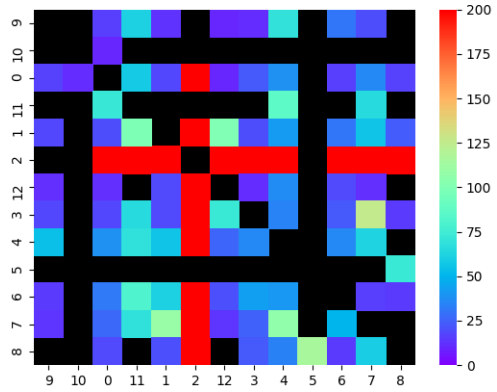

図 22 1000byte-1000msec のピア毎の RTT の 99%点

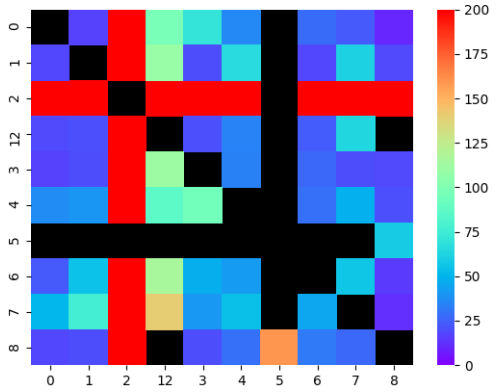

図 23 2000byte-1000msec のピア毎の RTT の 99%点

いため，上記の利点を生かし，遠隔地との P2P 性能の統計情報を積極的に蓄積することは新規性が大きいと考えられる．

参考文献

[1] Atlas console. https://atlas.ripe.net/about/. (Accessed on 02/03/2021).

[2] Introduce ourselves — inonius. https://inonius.net/charter-intro.html. (Accessed on 02/03/2021).

[3] Sindan project — sindan project. https://www.sindan-net.com/. (Accessed on 02/03/2021).

[4] Syncroom について. https://syncroom.yamaha.com/about/. (Accessed on 02/03/2021).

[5] Webrtc の通信状況をプログラマブルに判別するライブラリを作ってみた - qiita. https://qiita.com/monmee/items/77310482b0a6a034a887. (Accessed on 02/03/2021).

[6] プロジェクトについて — web videomark.

https://vm.webdino.org/about. (Accessed on 02/03/2021).
[7] User Datagram Protocol. RFC 768, August 1980.
[8] Harald T. Alvestrand. Overview: Real-Time Protocols for Browser-Based Applications. RFC 8825, January 2021.
[9] Romain Fontugne, Anant Shah, and Kenjiro Cho. Persistent last-mile congestion: Not so uncommon. In *Proceedings of the ACM Internet Measurement Conference*, pages 420–427, 2020.
[10] R. Jesup, S. Loreto, and M. Tüxen. WebRTC Data Channels. RFC 8831, January 2021.
[11] Sharad Agarwal Jacob R. Lorch. Matchmaking for online games and other latency-sensitive p2p systems. http://ccr.sigcomm.org/online/files/p315.pdf, August 2009.
[12] Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security. RFC 4347, April 2006.
[13] Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, April 2010.
[14] David Singer, HariKishan Desineni, and Roni Even. A General Mechanism for RTP Header Extensions. RFC 8285, October 2017.
[15] Randall R. Stewart. Stream Control Transmission Protocol. RFC 4960, September 2007.
[16] Chris Butcher Youngki Lee, Sharad Agarwal and Jitu Padhye. Measurement and estimation of network qos among peer xbox 360 game players.
[17] 島慶一. 家庭内ネットワーク遅延計測. https://iham.iijlab.net/latency/. (2020/7/20, WIDE プロジェクト"在宅オンライン活動の問題解決と快適化"提案資料).