※ 原則として、付録込みで最大 30 枚（Ａ４）。

卒業論文

論文題目 WebRTC による宅内環境間 P2P ラウンドトリップタイム計測

令和３年 ２月５日提出

指導教員 江崎 浩教 授

落合 秀也准教授

※指導教員は講師以上

電子情報工学科

03-190436 中川 紘輔

卒業論文

WebRTC による宅内環境間 P2P ラウンドトリップタイム計測

令和 3 年 2 月 5 日提出

指導教員　江崎 浩　教授
落合 秀也 准教授

電子情報工学科 03-190436 中川紘輔

概要

コロナウイルスの感染拡大に伴い，様々な Web アプリケーションが従来の社会機能の代替となりつつある．従来対面で行われるような仕事や教育の現場においてもリモートワークが増加し，それに従って家庭や生活圏でのネットワークの性能や品質を含めた課題が顕在化した．

そういった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 方式が採用されつつあり，宅内ネットワーク環境間 P2P 通信が行われ始めている．

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．

目次

1 序論 3
1.1 研究の背景 . . . 3
1.2 研究の目的 . . . 3
1.3 本論文の構成 . . . 4

2 リモート活動を支える宅内環境間 P2P 通信システム 5
2.1 クライアント・サーバーモデルの問題 . . . 5
2.2 P2P の有用性 . . . 5
2.3 P2P の課題点 . . . 5

3 WebRTC 概要 6
3.1 ICE . . . 6
3.2 シグナリング . . . 6
3.3 WebRTC の MediaStream . . . 6
3.4 WebRTC のデータチャネル . . . 6
3.5 統計情報に関連する実装 . . . 7

4 関連研究 9
4.1 WebDINO VideoMark . . . 9
4.2 SINDAN . . . 9
4.3 RIPE Atlas . . . 9
4.4 iNonius プロジェクト . . . 10
4.5 Home Area Measurement . . . 10
4.6 Persistent Last-mile Congestion: Not so Uncommon . . . 10

5 提案手法 11
5.1 機能要件 . . . 11
5.2 システムの概略 . . . 12
5.3 WebRTC による P2P 性能計測ツール . . . 13
5.4 データ収集サーバー . . . 15
5.5 可視化ツールの実装 . . . 15

6 実験 18
6.1 実行環境 . . . 18
6.2 実験 1（送信間隔を変化） . . . 18
6.3 実験 2（送信サイズを変化） . . . 19

7 考察 30
7.1 MediaStream と DataChannel のいずれが計測ツールに適しているかについて . . . 30

7.2 Web ブラウザ上で計測を行うことの長所・短所について 30
7.3 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 30

8 結論 31
8.1 今後の展望 31

1 序論

1.1 研究の背景

2020 年の新型コロナウイルスの感染拡大，およびその対処として各国で行われた外出自粛要請やロックダウンといった政策は，人々の生活に大きな変化をもたらした．ビジネスの現場ではテレワークが普及し，学校現場では授業や学会，研究活動などがオンラインで行われるようになった．あらゆる社会活動において，コロナ前の社会機能を可能な限り復元するために，Web 会議サービスやオンライン配信サービス等を導入する取り組みが活発化した．

このような過程で，既存の IT インフラを使用した Web アプリケーションには，通信性能的な限界が存在するということも明らかとなった．その一つとしてあげられるのが，宅内ネットワーク環境間での音声のリアルタイムストリーミングである．

このような通信は，主としてリモート合奏などの高精度同期が必要な音声ストリーミングを低遅延で行うことを目指して行われており，ヤマハ株式会社が開発する「Syncroom」等のリモート合奏アプリケーションが先行事例として挙げられる．しかしながら，音声のリアルタイムストリーミングにおいて使用される従来の TCP/IP 通信は，本来リアルタイム性を担保して設計されているものではない．

この影響によって生じる遅延は，音楽分野で要求される同期精度という観点において，無視できない数値となる．したがって，Syncroom のようなアプリケーションのユーザーは，常に下位レイヤの実装（無線 LAN か有線 LAN か，ネットワークの経路や帯域幅，バッファリングの大きさ）を縦断的に意識して遅延を抑制する必要に迫られる．

また，ある宅内環境同士で得られる通信性能も，一定水準の性能が担保されるものではなく，実際に環境を用意してプログラムを動かさないとわからないのが現状である．また，同じ環境でも時間帯によって性能が変動する可能性がある．これはサービス運用者が品質向上を行うにあたり，まず多大なデータ蓄積を行う必要性に迫られることを意味する．

この現状を打開するためには，宅内ネットワーク環境間を結ぶ P2P 通信の性能を広範に計測し，データを収集することが問題解決の第一歩として重要である．

1.2 研究の目的

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．

1.3 本論文の構成

本論文の構成は，まず第 2 章で，本研究の背景について具体的な事例紹介を交えて述べる．第 3 章では，本研究の核となる技術である WebRTC について概説する．第 4 章では関連研究について述べる．第 5 章では，本研究の提案手法を述べる．第 6 章では，実験の詳細および結果について述べる．第 7 章では実験結果に対して考察を行い，第 8 章で本研究の結論および今後の展望を述べる．

2　リモート活動を支える宅内環境間 P2P 通信システム

2.1　クライアント・サーバーモデルの問題

クライアント・サーバーモデルによる通信は，あらゆる Web サービスにおいて主流であるものの，リアルタイム性を担保して設計されているとは言い難く，パケットをストリーミング送受信する場合には，通信路の中間に存在するバッファリング処理やパケット分割などの様々な影響により，安定的に低遅延を実現することは難しい．
また，サーバーを中継する通信は必然的に経路長が長くなるため，低遅延通信が必要なシステムでは最適な通信方式とは言い難い．

2.2　P2P の有用性

上記のような背景により，可能な限り「音声品質を維持しつつ，遅延を減らす」ことを目指し，従来主流であった「クライアントサーバモデル」のシステムではなく，「 P2P モデル」のシステムを作る取り組みがなされている．以下その一事例について紹介する．

2.2.1　Yamaha「Syncroom」

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．

2.3　P2P の課題点

宅内環境〜宅内環境の P2P 通信性能に関しては先行研究が乏しく，実際にこのようなシステムがどの程度の遅延の下で使用されているかは未知数である．
先述の Syncroom のような P2P 音声ストリーミングソフトの場合，ある一つのノードの遅延が非常に大きくなってしまう等の事象が発生した場合に，参加者全体が遅れて聞こえてくる音声によって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．
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アンケート調査を実施したり，統計情報を収集したりすることが多い．
P2P 通信を用いたシステムが今後増加した場合，より体系的な統計情報を収集することがスムーズな運用には不可欠であると考えられる．

3 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．

3.1 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．[12]

3.2 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．

3.3 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．[13]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．

3.4 WebRTC のデータチャネル

WebRTC のデータチャネルは 2021 年 1 月現在，RFC で標準化過程にカテゴライズされている．
2021 年 1 月現在で最新の RFC 8831[10] に記載されている具体的な仕様は以下．

3.4.1 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．[11]

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．[14]

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．

3.4.2 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．

3.5 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．

3.5.1 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．

3.5.2 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]

4　関連研究

以下で，宅内ネットワークの性能計測と関連する先行研究について概説する．

4.1　WebDINO VideoMark

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．
また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．

4.2　SINDAN

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おり，具体的には，以下の２つの研究目標を掲げている．

- ネットワーク運用モデルの体系化と評価手法の確立：ユーザの利用環境におけるネットワーク状態を評価する手法を，アクセス回線の種別や利用するインターネットプロトコルといったネットワークの運用形態毎に定義する．実験的手法を通して，ネットワーク運用のモデル化に必要な階層を定義した上で，各階層で発生しうる障害を体系化する．
- ネットワーク状態の記述手法の定義と標準化：ユーザ側からの計測するネットワーク状態を単純な接続性だけでなく，アプリケーション毎の到達性も含めて定義する．ネットワーク接続性記述手法を定義・標準化し，ユーザー側のネットワーク状態の定量的な可視化手法を開発する．

4.3　RIPE Atlas

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．
RIPE Atlas はオープンソースのプロジェクトであり，収集された統計情報は Internet Map として一般公開されている．

4.4 iNonius プロジェクト

従来，ある組織の構成員の宅内インターネット環境調査はアンケート等の手法により行われていたが，COVID-19 による影響でテレワークが拡大している中，このような調査がより効率的・迅速に行える環境を整備する必要性が生じている．
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].

4.5 Home Area Measurement

在宅オンライン活動の拡大で，宅内のネットワーク環境を計測する必要性が広く生じている．しかし，専門的知識がない人が ping 等の方法を用いて宅内ネットワークを計測する事は困難である．
これを Web 上で簡単に行うために，このプロジェクトでは

$$(\text{ブラウザから計測サーバーまでの遅延}) - (\text{計測サーバーから最遠の } tracepath) \tag{1}$$

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 [15].

4.6 Persistent Last-mile Congestion: Not so Uncommon

ラストワンマイルとはインターネットサービス加入者の自宅から最寄りの通信拠点までの接続のことを指し，この部分における性能がブロードバンド接続の品質を大きく左右するということが一般的に知られている．
[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．

5　提案手法

今後は上記のオープンソースのツールを元に，以下のような要件を満たしたツールを製作する．

5.1　機能要件

5.1.1　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．

5.1.2　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．

5.1.3　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．

5.1.4　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．

5.1.5　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.

5.1.6　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.

5.2　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 と図 2 と図 3 に示す.

本手法の全体像としては，1 に示すように，

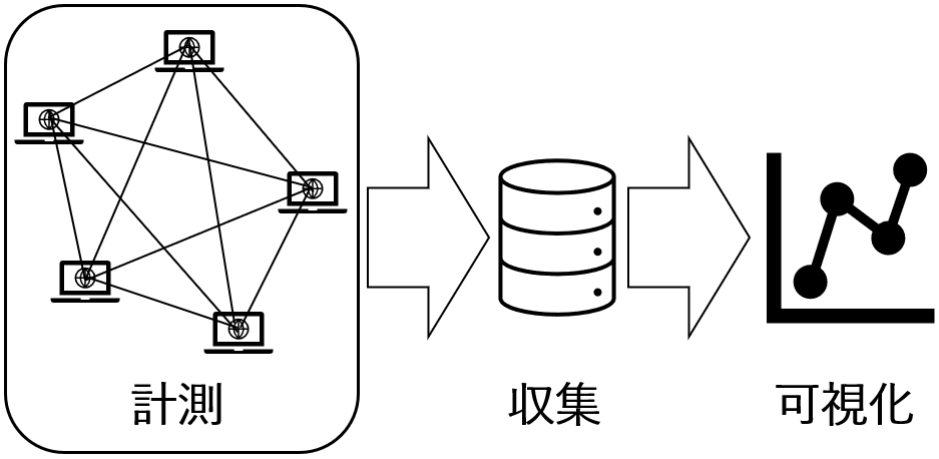


図 1　提案手法の概念図

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，図 2 で示されている通り，「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，図 3 で示されているような概念である．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.

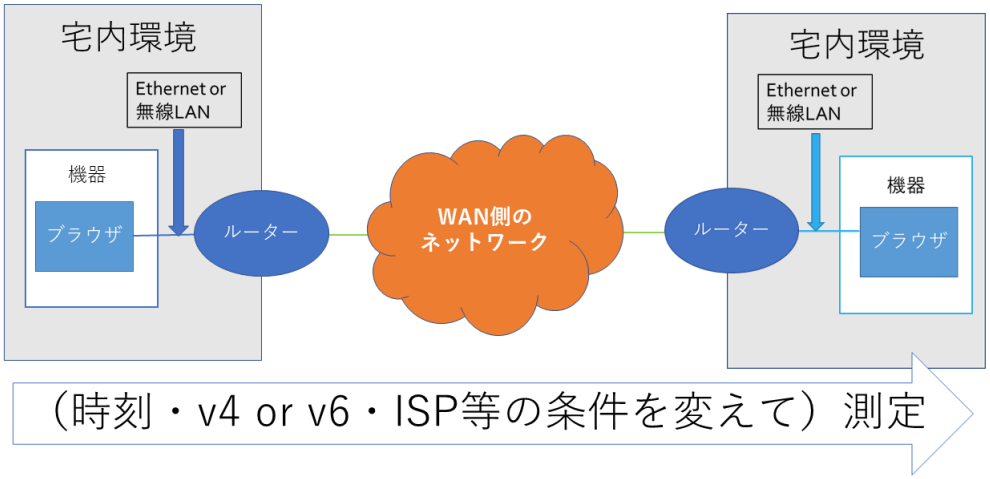


図 2　ピア同士の接続イメージ

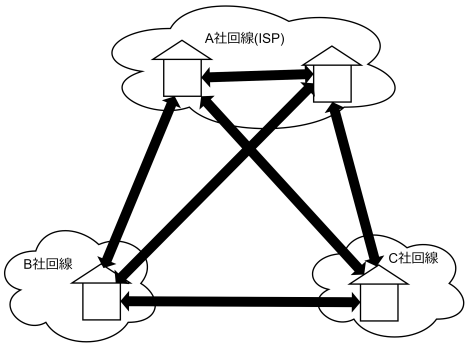


図 3　多ノードのフルメッシュ接続イメージ

5.3　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.

5.3.1 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．

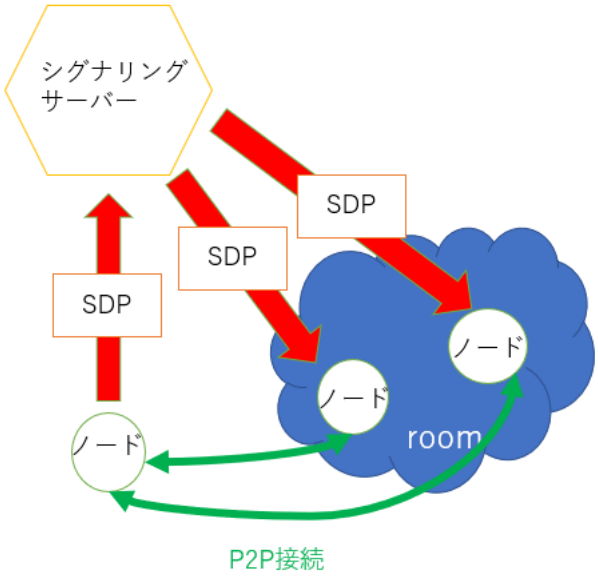


図 4　シグナリングサーバーの概念図

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．

5.3.2 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以下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



図 5　計測ツールの Web フロント

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 6 に示す時系列で動作する．

5.4　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 7 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．

5.5　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

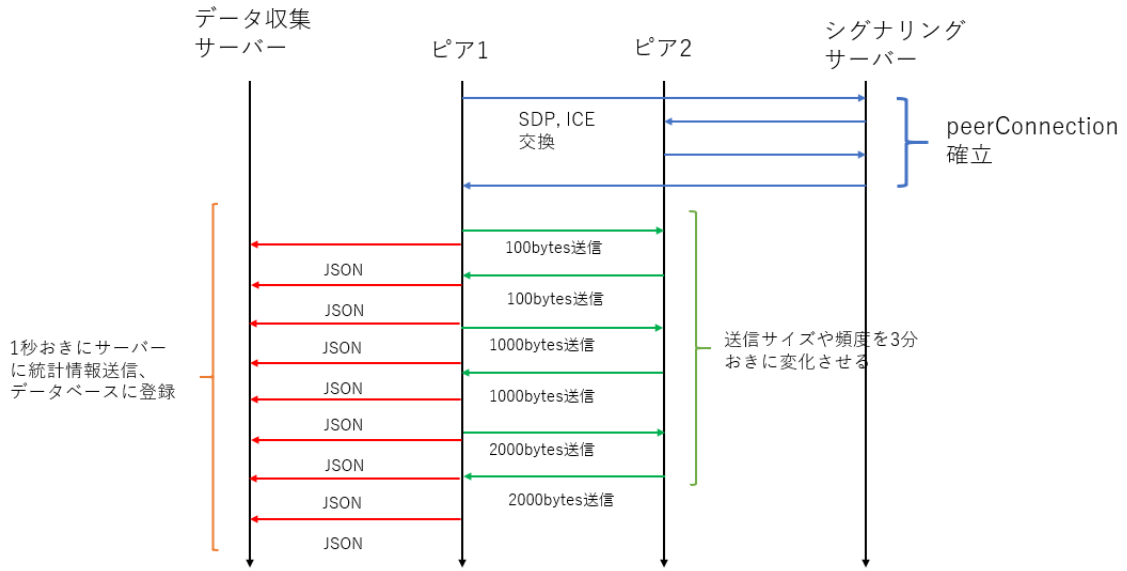


図 6　計測ツール全体の動作

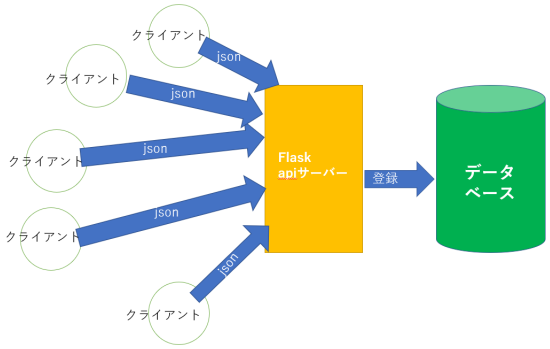


図 7　データ収集サーバーの図

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

表 1　色と RTT の対応

色	RTT
青	$RTT <= 20[ms]$
シアン	$20[ms] < RTT <= 50[ms]$
緑	$50[ms] < RTT <= 100[ms]$
橙（太線）	$100[ms] < RTT <= 200[ms]$
赤（太線）	$200[ms] < RTT$

表 2　色とノードの接続情報の対応

色	接続方法
青	有線 LAN
赤	Wi-Fi
白	その他・未回答含む

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，以下のような 2 種類の可視化を行う．

5.5.1　有向グラフによる可視化

有向グラフによる可視化では，まずフルメッシュに張った有向グラフを作成し，RTT 値に応じて以下のような着色を行う．

- P2P 性能によるエッジの色付け：各ピア同士の RTT の平均値，および RTT 値の 99% 点の値を取得し，その値に応じてエッジの色を変更する．
- P2P 性能によるノードの色付け：各ノードが取得した RTT の平均値，および RTT 値の 99% 点の値を取得し，その値に応じてノードの色を変更する．

99% 点という指標を採用したのは，後述の実験結果のヒストグラムを見るとわかるように，ごく少数 (1-10 個オーダー) のデータで極めて大きな RTT 値を記録するという現象が観測されたためである．ナイーブに外れ値である最大値で議論をしても，実際の通信品質との関連性が低くなってしまうと考えられるため，今回は外れ値を排除するために 99% 点とした．
また，通信性能をランク分けをし，ランクに応じて各ノードやエッジに色付けすることにより，各ノード，およびピア間の P2P 接続性能の直感的な理解を目指した．また，通信性能が 100[ms] より大きいピアのエッジは太線で表示することで，問題がある通信の視認性を高めた
表 1 に RTT による各ノードの色分け，表 2 に端末接続方法による各ノードの色分けを示す．

5.5.2　ヒートマップによる可視化

また，ヒートマップによる可視化も行った．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．

6 実験

6.1 実行環境

- Web サーバー：Ubuntu 18.04/Apache2(https 接続)
- 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　江崎・落合研究室
- クライアントが使用する 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．

6.2 実験 1（送信間隔を変化）

6.2.1 実験概要

- 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）
- 条件：送信サイズ 100[バイト]，送信間隔 100,300,1000[msec]
- クライアントが使用する Web ブラウザ：Google Chrome

の条件で計測を行った．

6.2.2 実験結果

図 8, 図 9, 図 10 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．
図 14, 図 17 を見ると，平均値で見た場合青色の 8 番のノードが，99% 値では黄色に分類されるという現象が発生している．これは，平均的な性能と悪化したときの性能のばらつきが相対的に大きいことを意味しており，通信品質の変動が激しい可能性が高いと考えられる．
また，100byte-1000msec のみ接続ノード数が大きい事象に関しては，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 8.1.2 を参照）
送信間隔 1000[ms] の図は，相対的に青や緑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．
可視化した結果としては，赤色で表示されている 3 番のノードは通信品質が極めて悪く，P2P 通信を使用した音声ストリーミングを行うことが困難であると考えられる．また，図 17 を見ると 3 番と接続している 13 番は橙色で表示されているが，3 番以外のノードとは比較的安定した通信を行うことができており，3 番の通信

品質の影響を大きく受けた結果となっていると考えられる．
図 11 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．
また，図 12 で各ノードの接続方法を可視化した（青が有線，赤が Wi-Fi）．しかし，接続方法の差異によって通信品質が影響を受けていることを確認することはできなかった．
図 19-24 はヒートマップによる可視化である．図 19 と図 20，図 22 と図 23 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．

6.3 実験 2（送信サイズを変化）

6.3.1 実験概要

- 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）
- 条件：送信サイズ 100,1000,2000[バイト]，送信間隔 1000[msec]

の条件で計測を行った．

6.3.2 実験結果

図 25, 図 26, 図 27 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．
送信サイズ 100[bytes] の図は，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．
可視化した結果としては，すべてのグラフにおいて橙・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 通信を使用した音声ストリーミングを行うことが困難であると考えられる．
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 28 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．
また，図 29 で各ノードの接続方法を可視化した（青が有線，赤が Wi-Fi）．しかし，接続方法の差異によって通信品質が影響を受けていることを確認することはできなかった．
図 36-41 はヒートマップによる可視化である．図 36 と図 38，図 39 と図 41 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．

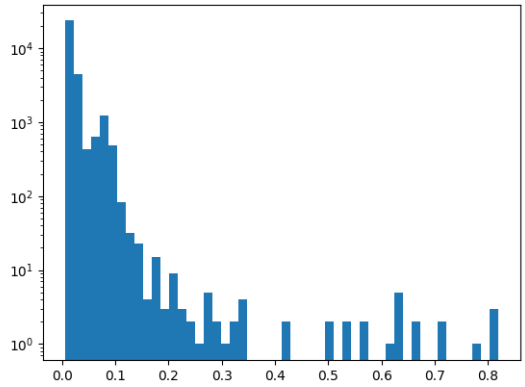


図 8　100byte-100msec の全取得 RTT のヒストグラム

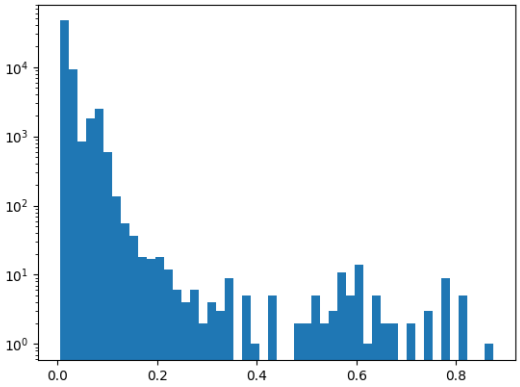


図 9　100byte-300msec の全取得 RTT のヒストグラム

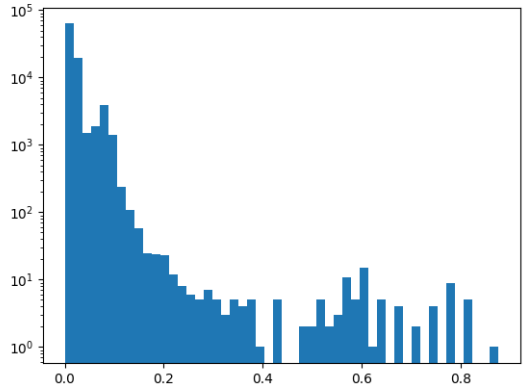


図 10　100byte-1000msec の全取得 RTT のヒストグラム

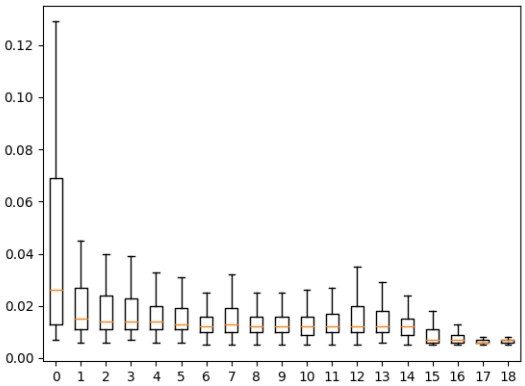


図 11　100byte-1000msec の RTT の時間変化（9 分毎）

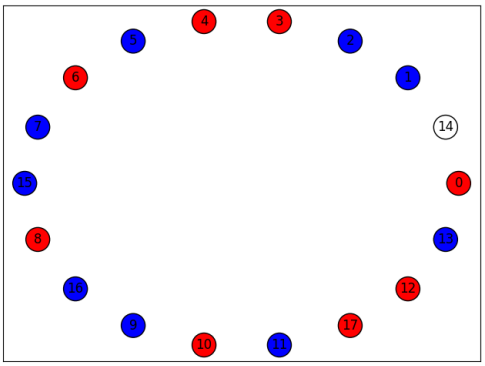


図 12　100byte-1000msec の各ノードの接続方法

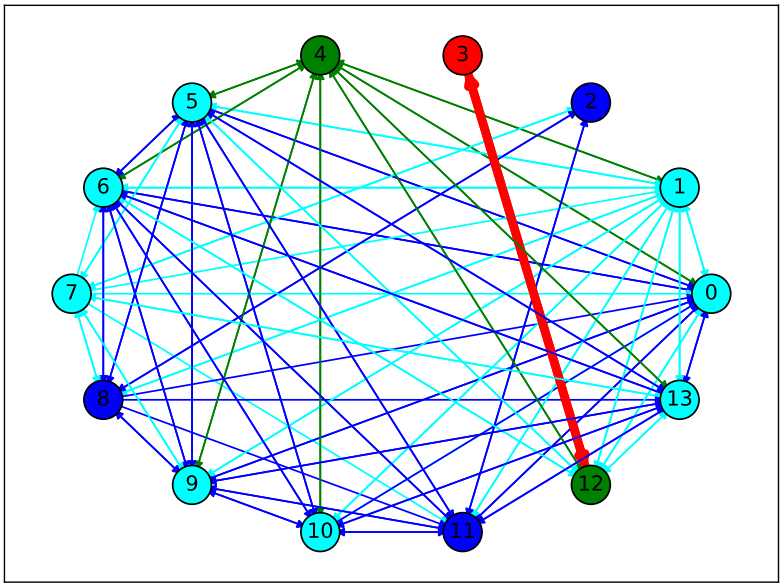


図 13　100byte-100msec のピア毎の RTT 平均値

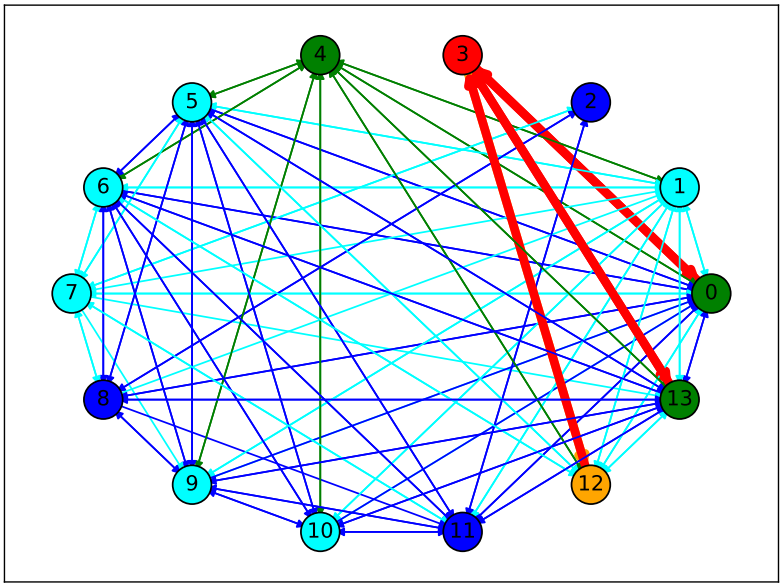


図 14　100byte-300msec のピア毎の RTT 平均値

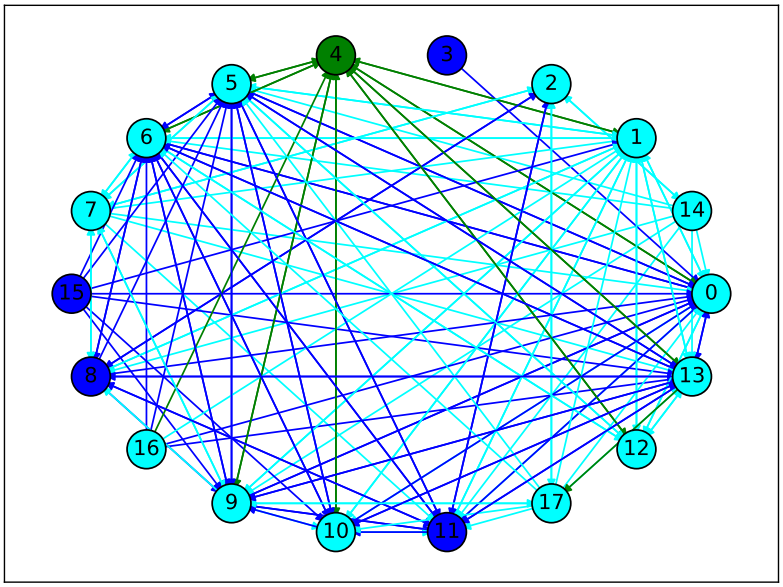


図 15　100byte-1000msec のピア毎の RTT 平均値

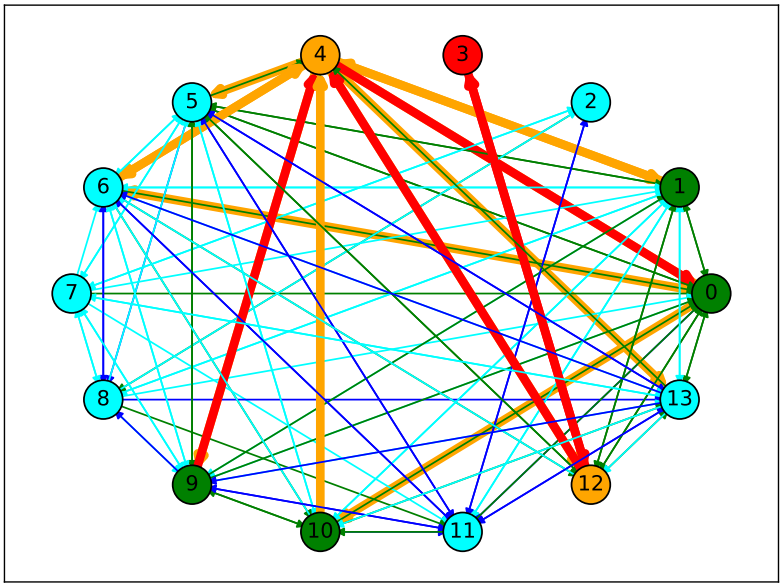


図 16　100byte-100msec のピア毎の RTT の 99% 点

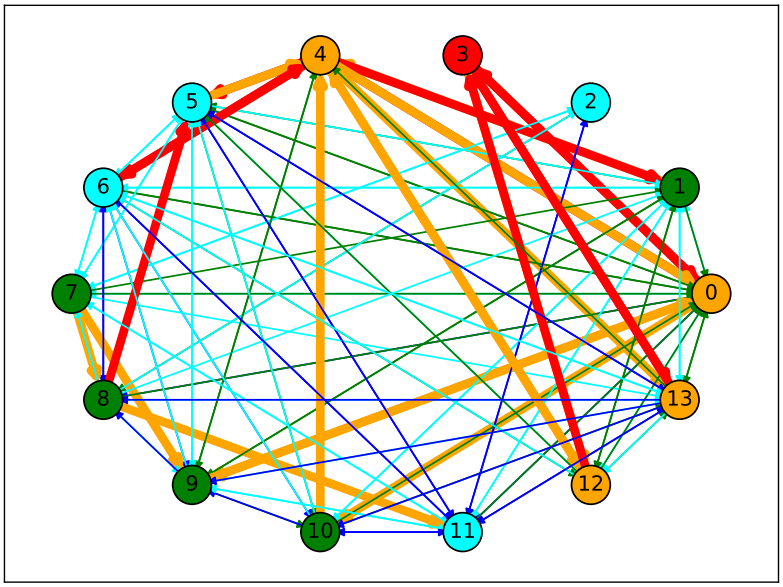


図 17　100byte-300msec のピア毎の RTT の 99% 点

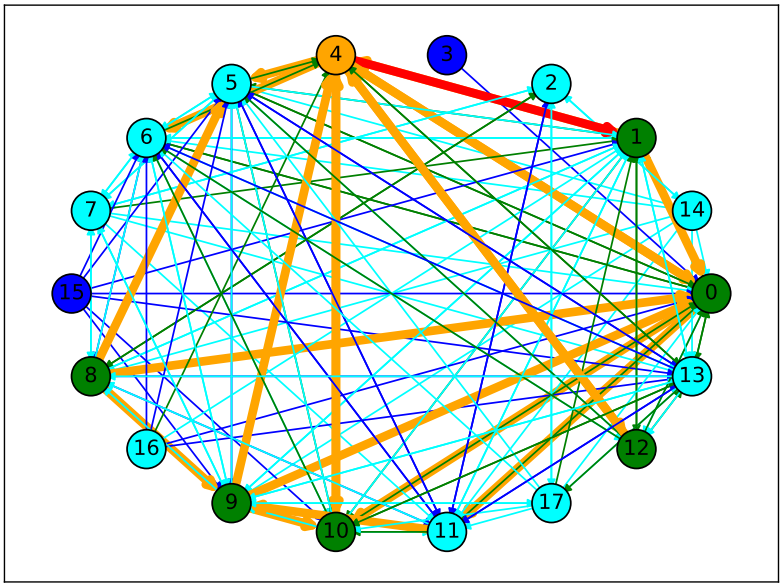


図 18　100byte-1000msec のピア毎の RTT の 99% 点

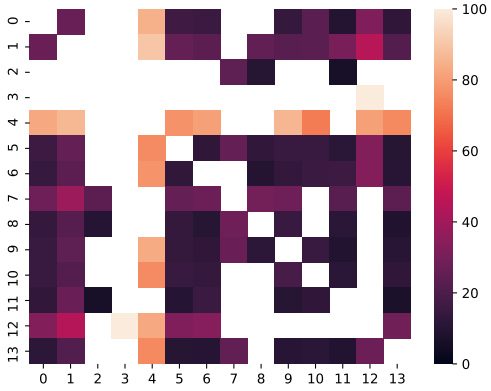


図 19　100byte-100msec のピア毎の RTT 平均値

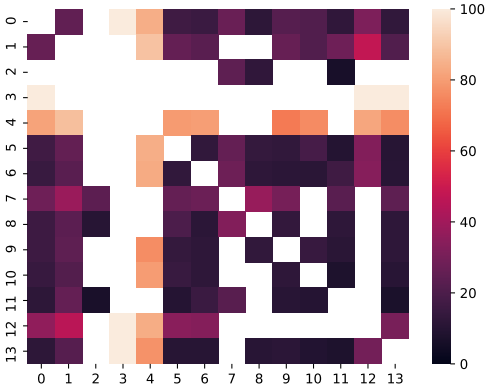


図 20　100byte-300msec のピア毎の RTT 平均値

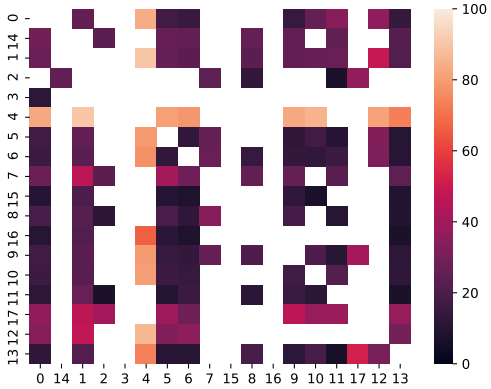


図 21　100byte-1000msec のピア毎の RTT 平均値

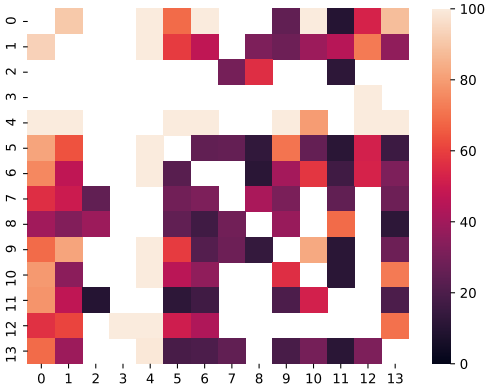


図 22　100byte-100msec のピア毎の RTT の 99% 点

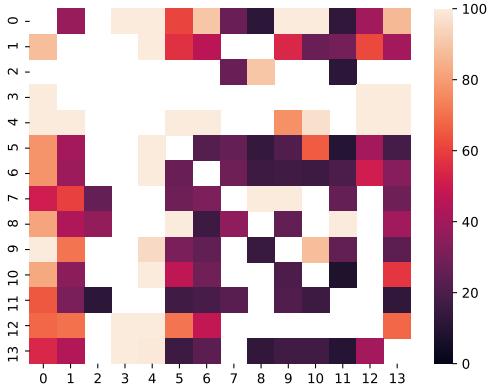


図 23　100byte-300msec のピア毎の RTT の 99% 点

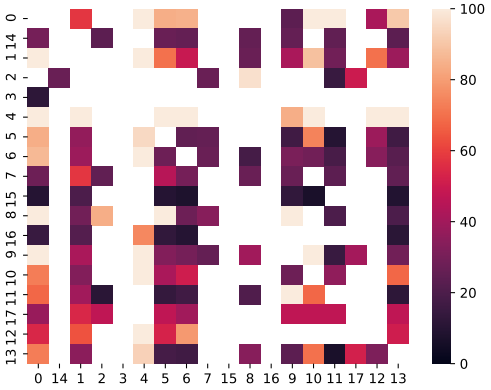


図 24　100byte-1000msec のピア毎の RTT の 99% 点

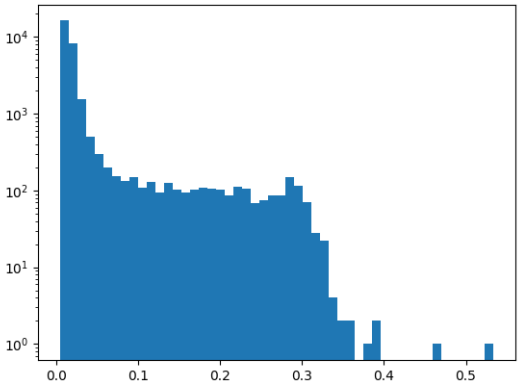


図 25　100byte-1000msec の全取得 RTT のヒストグラム

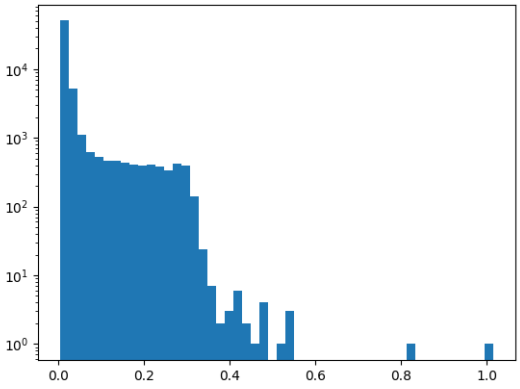


図 26　1000byte-1000msec の全取得 RTT のヒストグラム

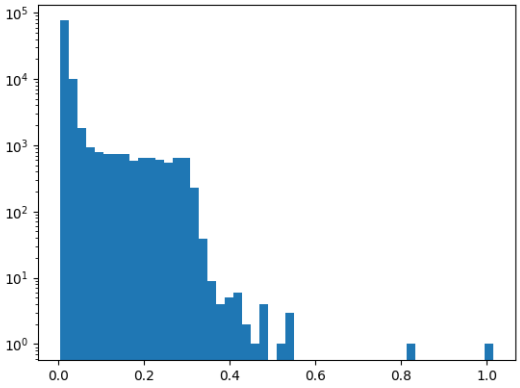


図 27　2000byte-1000msec の全取得 RTT のヒストグラム

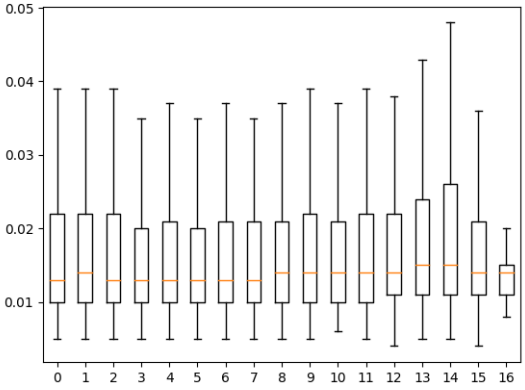


図 28　100byte-1000msec の RTT の時間変化（9 分毎）

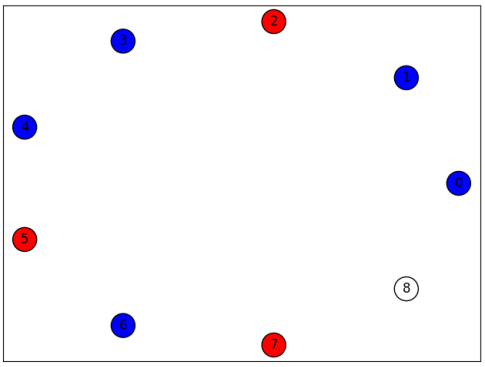


図 29　100byte-1000msec の各ノードの接続方法

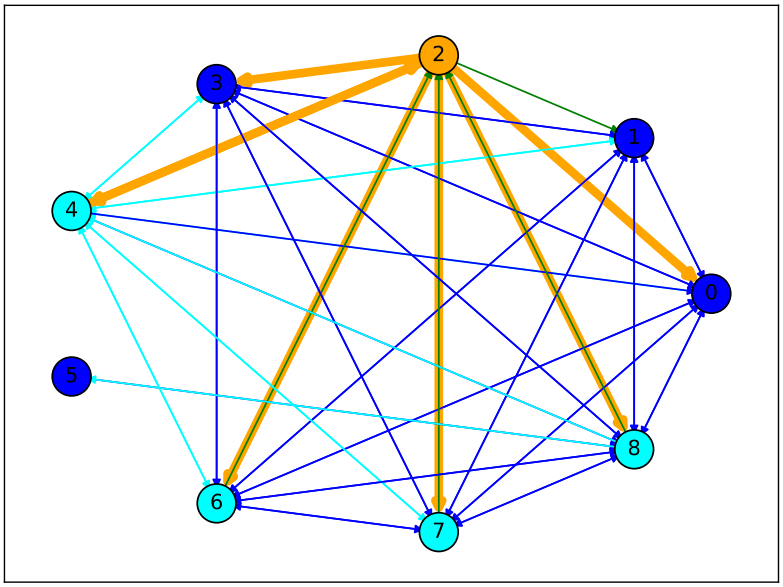


図 30　100byte-1000msec のピア毎の RTT 平均値

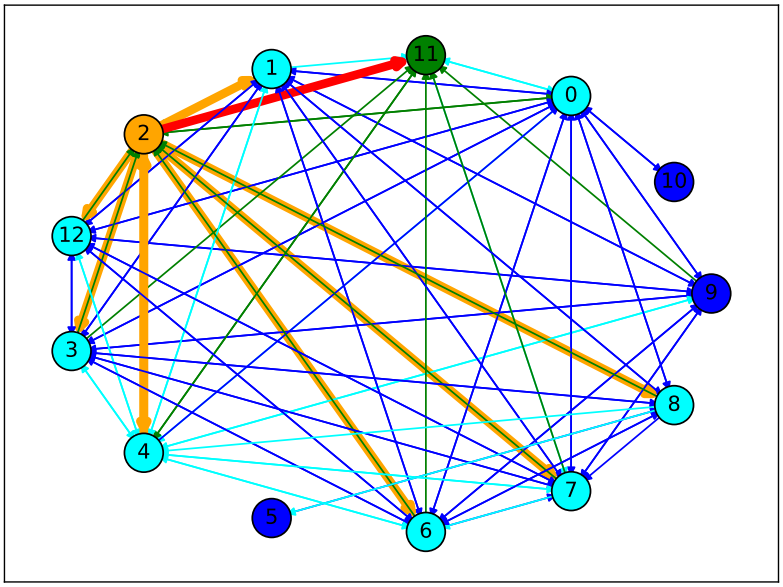


図 31　1000byte-1000msec のピア毎の RTT 平均値

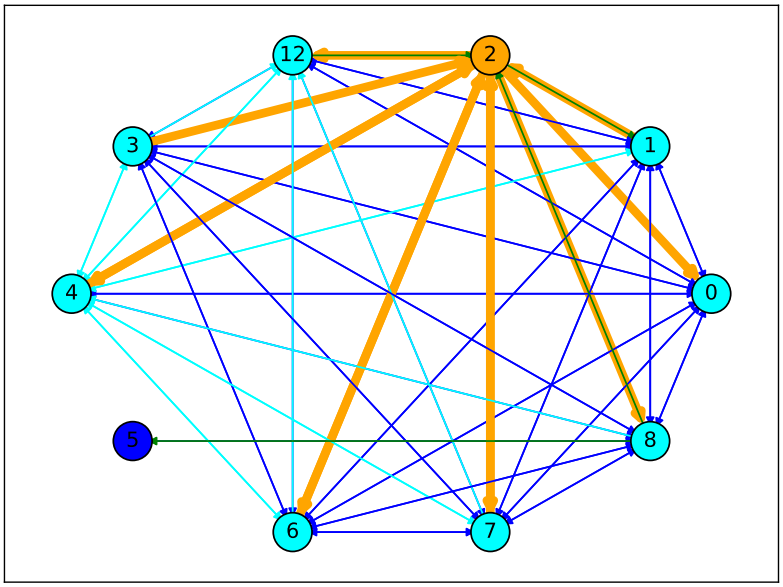


図 32　2000byte-1000msec のピア毎の RTT 平均値

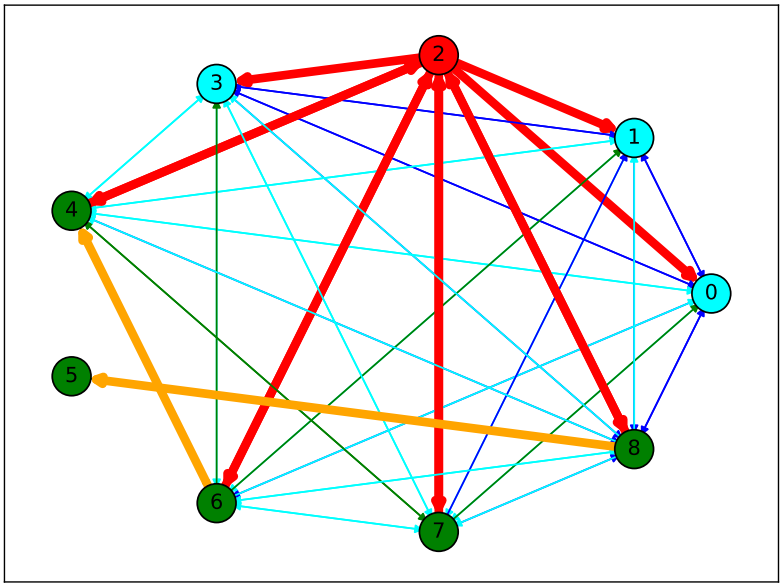


図 33　100byte-1000msec のピア毎の RTT の 99% 点

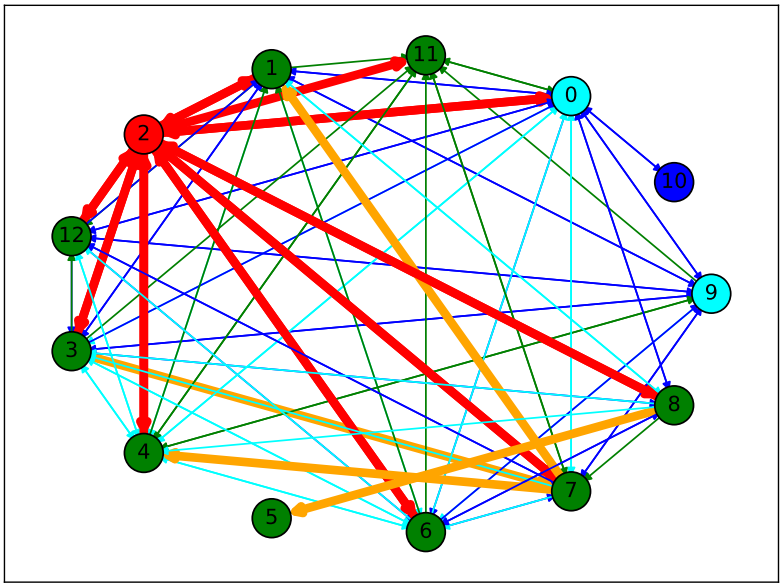


図 34　1000byte-1000msec のピア毎の RTT の 99% 点

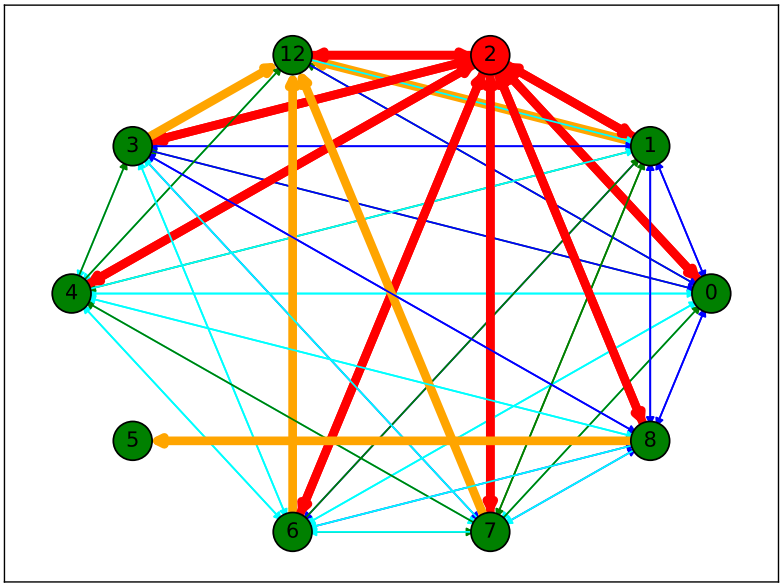


図 35　2000byte-1000msec のピア毎の RTT の 99% 点

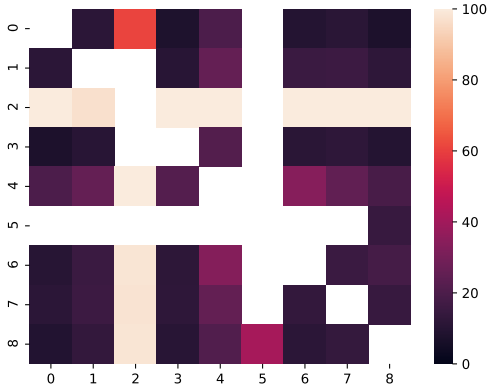


図 36　100byte-1000msec のピア毎の RTT 平均値

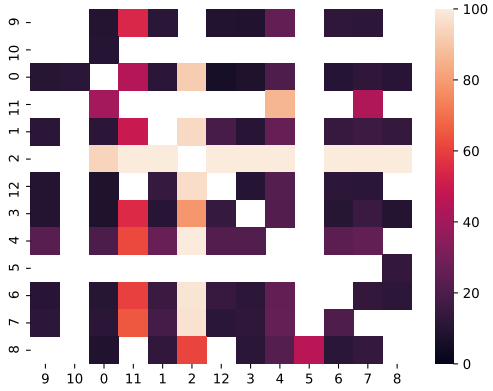


図 37　1000byte-1000msec のピア毎の RTT 平均値

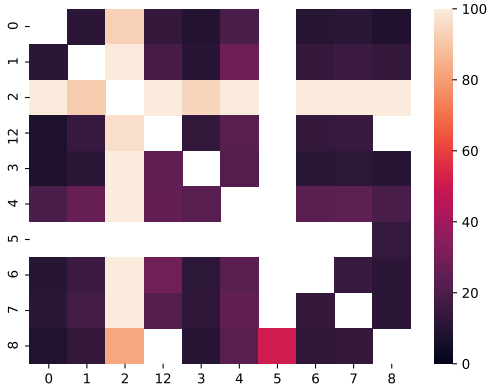


図 38　2000byte-1000msec のピア毎の RTT 平均値

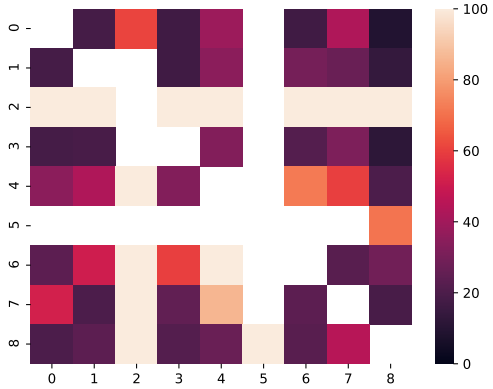


図 39　100byte-1000msec のピア毎の RTT の 99% 点

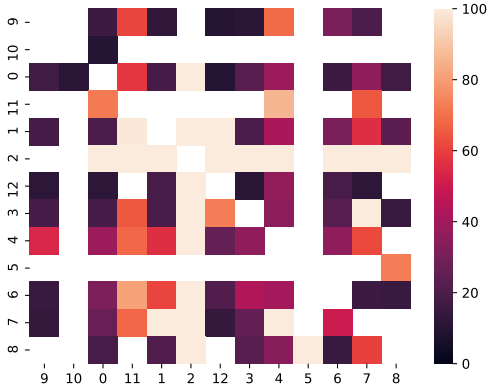


図 40　1000byte-1000msec のピア毎の RTT の 99% 点

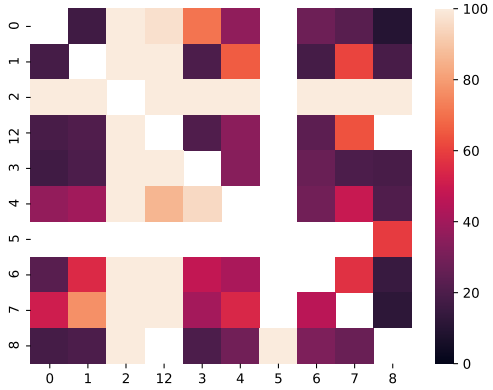


図 41　2000byte-1000msec のピア毎の RTT の 99% 点

7　考察

7.1　MediaStream と DataChannel のいずれが計測ツールに適しているかについて

本論文における最大の問題意識として，リモートのノード同士のリアルタイムストリーミングにおける通信の乱れというものを挙げている以上，ストリーミング性能そのものについて計測可能な前者のほうが問題意識により見合った計測であるといえる．
しかしながら，実際の計測において，前者の問題点が数多く明らかとなった．まずは，Web カメラを使用するストリーミング性能計測ツールは，物理的な環境依存性が極めて強く，広範な計測には不向きであるといえる．また，ストリーミングによる帯域の圧迫や CPU, GPU 負荷も大きく，接続するノードを増やせば増やすほど各ノードに要求されるハードウェアの性能も大きなものとなってしまう．
一方，ストリーミングではなくデータチャネルを用いてデータを取得する方法では，ハードウェアに求められる要件が緩く，より広範に計測を行うことが可能である．またデータ量も可変とすることができ，計測の拡張性は大きいと考えられる．

7.2　Web ブラウザ上で計測を行うことの長所・短所について

P2P 通信を用いたソフトウェアは先述の通り有用性が高い反面，匿名性やセキュリティ，帯域圧迫といった問題から歴史的に賛否が分かれている分野である．過去には P2P 通信による違法なファイル共有が横行し，ソフトウェアの開発者が逮捕される（のちに無罪確定）事案や，コンピューターウイルスが拡散される事案が発生しており，現在に至るまで必ずしも市民権を得られている技術であるとは言い難い事情がある．
このような経緯から，現状においても P2P ファイル共有ソフトウェアの使用を規制しているネットワークは多く，P2P というワードそのものに抵抗感を抱く人も多い．したがって，本論文のような計測を専用アプリケーション等を作成し，多くの人に広範にダウンロードをしてもらったうえで行うのは（違法なファイル共有は行わないとしても）ハードルが高い．
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．

7.3　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．

8 結論

本研究では，WebRTC による多ノード間 P2P 通信性能測定ツールを作成し，実際に統計情報を収集したうえで可視化を行った．その結果，通信性能が悪いノードを視覚的に見つけることが可能であり，具体的な問題解決に貢献する手法となりうる可能性があるという点において，意義が大きいと結論付けられる．

8.1 今後の展望

8.1.1 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．

8.1.2 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．

8.1.3 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．

8.1.4 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] Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security. RFC 4347, April 2006.

[12] Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, April 2010.

[13] David Singer, HariKishan Desineni, and Roni Even. A General Mechanism for RTP Header Extensions. RFC 8285, October 2017.

[14] Randall R. Stewart. Stream Control Transmission Protocol. RFC 4960, September 2007.

[15] 島慶一．家庭内ネットワーク遅延計測．https://qiita.com/monmee/items/77310482b0a6a034a887. (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).

謝辞

本論文の執筆にあたり，多くの方のご助力，ご指導を賜りましたことをこの場を借りて感謝申し上げます．
まず，研究を進めるにあたり必要となる予備知識や問題設定のための背景知識について，多くのご指導を賜りました江崎浩教授，落合秀也准教授，塚田学准教授に感謝申し上げます．
また，SDM 勉強会で WebRTC の技術的な知識や研究の進捗管理について多くのご指導を賜りました shin 氏，ino 氏，sone 氏に感謝申し上げます．
Syncroom 等サービスの運用を通して得られた問題意識を共有していただき，問題設定のきっかけを作ってくださいましたヤマハ株式会社の皆様に感謝申し上げます．
測定ツールの動作確認，および実際の計測にご協力賜りました多くの皆様に感謝申し上げます．