Vol.2012-AVM-76 No.3
2012/2/23

ローカル・ラグ制御機能とログ同期機能を持つ音響サーバの開発

竹森幸輝[†1]　前田佳奈[†1]　岩原正典[†1]
片桐滋[†1]　大崎美穂[†1]

遠隔同時コラボレーションの重要性に注目して，我々は，その典型的な事例の一つである遠隔合奏課題に伴う遅延の問題の研究を行ってきた．本稿は，この遠隔合奏課題にローカル・ラグの概念を適用する正確な実験を行うことを目指して開発した，遠隔合奏を支援するための新しい音響サーバを紹介するものである．本サーバは特に，複数の遠隔サーバ間で時刻同期を実現し，また録音ファイルにも共通した同期スタンプを押印する機能を持っている．実験を通して，本サーバの基本的性能を明らかにし，本サーバが遠隔合奏実験のプラットフォームになり得ることを示す．

Development of Acoustic Server Having Local Lag Control and Log Syncronization

Koki Takemori,[†1] Kana Maeda,[†1]
Masanori Iwahara,[†1] Shigeru Katagiri[†1]
and Miho Ohsaki [†1]

Focusing on the importance of synchronized remote collaboration, we have investigated a time lag problem that exists in the remote ensemble performed over the public digital network, which is a typical example of remote synchronized collaboration. To conduct accurately controlled experiments using the local-lag concept in this remote ensemble task, we develop in the present study a new acoustic server system for supporting remote ensembles, which has a function of time stamp synchronization among remote servers and also has a function of setting a time stamp that is common to the recorded data files at remote sites. Through an elaborative measurement experiment, we demonstrate that the developed server can be a reliable platform for remote ensemble experiments.

1. はじめに

我々は，最近急速に大きな関心を集めている遠隔共同同時作業の典型的な事例として遠隔合奏課題を採り上げ，遠隔地までの通信遅延と同量の遅延をローカル側の演奏音に付加して再生するローカル・ラグ法という遅延対策法[1]の，遠隔合奏に対する効果について調査してきた[2]．

遠隔合奏を支援するシステムとしては，これまでにもJackTripやNETDUETTOなどにみられるような様々なシステムが開発されてきた[3][4]．しかし，ローカル・ラグの制御実験をするため，我々は独自の遠隔合奏支援システムを開発し，先の実験を行ってきた．実験の結果，遅延が及ぼす遠隔合奏への悪影響はローカル・ラグの追加によって一定程度軽減されることが明らかとなった．しかし，そこで用いられた遠隔合奏支援システムは，2地点の合奏にしか対応できず，また実験結果の分析に用いるログファイルの時刻同期の機能を持っておらず，演奏地点の増加やより精度の高い実験を行うためには不十分であった．

こうした背景に基づき，我々は，高精度なローカル・ラグ制御を伴う遠隔合奏評価研究に用いる新しい遠隔合奏支援システムの開発を行ってきた．本システムは，各地点ごとに1台ずつ配置される音響サーバと，任意の地点に配置される1台の同期制御サーバから構成される．特に新支援システムで中心的な役割を果たす音響サーバは，演奏音の入出力を行うサーバで，音データの送受信や録音といった従来の機能に加え，ローカル・ラグを用いて3地点以上を接続する機能や，各音響サーバで記録される音声ファイルに同期がとられた共通のタイムスタンプを押印する機能を持つ．コンピュータ・ネットワークを経由する遠隔合奏においては，MIDI規格の電子楽器を用いた合奏などの方が扱いやすいとも思われる．しかし我々は，遠隔合奏以外の様々な遠隔同時作業への適用性を保つため，マイク収音されたオーディオ・データを扱うことができる音響サーバを開発することとした．本論文は，この新しい遠隔合奏支援システムの詳細と，システム性能の基本である，音響サーバの時刻同期制御の精度分析を中心とした性能評価結果について報告するものである．

2. ローカル・ラグを用いた遠隔合奏

離れた2地点間で遠隔合奏を行ったときの図を図1に示す．それぞれの地点における時

†1 同志社大学大学院　工学研究科
Graduate School of Engineering, Doshisha University

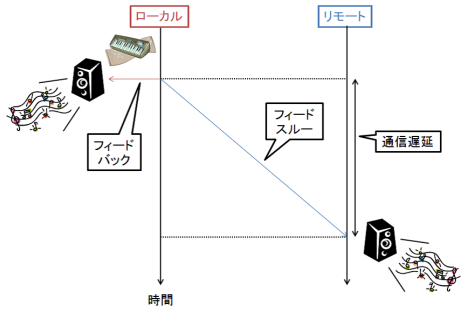

図 1 通常の遠隔合奏.

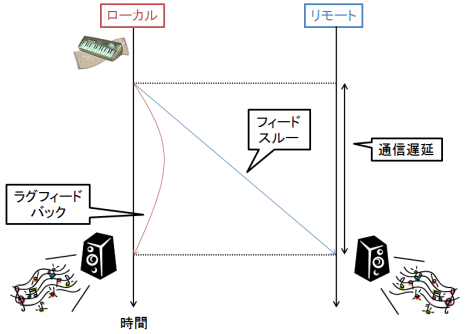

図 2 ローカルラグを用いた遠隔合奏.

間の経過は，縦の時間軸によって表現される．演奏者自身が聴く演奏音をフィードバックと呼び，通信処理によって合奏相手に届けられる演奏音をフィードスルーと呼ぶ．図 1 に示されているように，ローカル側の演奏者の演奏音は，通信遅延を伴ってリモート側の演奏者に届けられる．リモート側の演奏者は，遅れて聴こえるフィードスルーのタイミングに合わせるべく，本来演奏すべきタイミングより遅れて演奏することになる．これを繰り返していくと，次第に演奏の遅れが増大し，やがて演奏が破綻してしまう．

この問題に対して，各地点の演奏者が聴く音の同期を確保し，次に演奏すべきタイミングを計るための起点を揃えることで，演奏の破綻防止を図る遅延対策手法がローカル・ラグである[2)]．図 2 に，ローカル・ラグ制御を用いた遠隔合奏の原理を図解する．ここで，フィードバックに通信遅延分のラグを付加したものをラグフィードバックと呼ぶ．図 2 に示されている様に，本来なら即座に再生されるフィードバックに対し，フィードスルーにかかる通信遅延と同量の遅延を付加して再生することで，各地点の演奏者が聴く音の同期が確保される．これによりローカル側とリモート側の両演奏者は，次に演奏するタイミングを同一の起点から計ることができ，演奏の破綻を防ぐ効果が期待できる．

3. 新しい遠隔合奏支援システム

3.1 システムの構成

図 3 に，開発した遠隔合奏支援システムの，3 地点間接続時における構成例を図示する．本システムは，音響サーバと同期制御サーバの 2 種類のサーバから構成される．音響サーバは，遠隔合奏に関する主要な機能を持つサーバで，音の入出力機能，ローカル・ラグ制御機能，各地点共通のタイムスタンプを押印するための時刻同期機能，共通タイムスタンプを伴

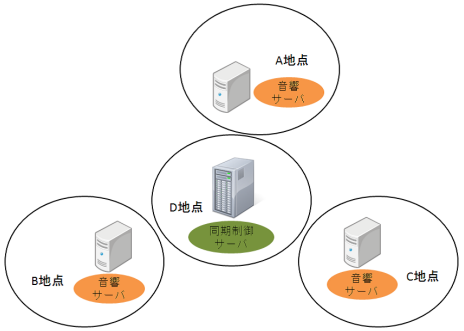

図 3 システム構成例.

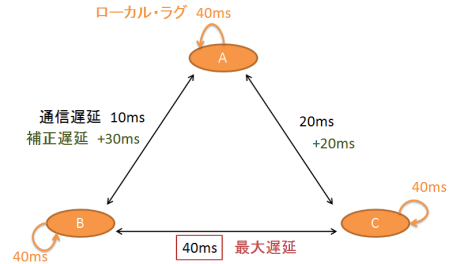

図 4 3 地点間以上におけるローカル・ラグを用いた遠隔合奏.

うファイル書き出し機能（音データ録音機能）などを持つ．同期制御サーバは，音響サーバを補助する役割を持つサーバで，音響サーバ間の接続や，ローカル・ラグ制御機能，時刻同期機能などの補助を行う．同期制御サーバは任意の地点に 1 台だけ必要であり，音響サーバは各地点に 1 台ずつ設置する必要がある．同期制御サーバと音響サーバを 1 台のマシン上で動作させることもできる．

3.2 独自タイマ

Linux において同期処理に用いることができる時刻情報として一般的なものに，カーネルが管理するシステムクロックが挙げられる．システムクロックは tick と呼ばれる一定の周期で動作し，時を刻む．この tick 単位で刻まれるシステムクロックを時刻情報として用いることで同期処理が実現できるが，この時刻情報は多少複雑なデータ構造を持っており，時刻同士の単純な足し引きなどの計算処理にもある程度手順を踏む必要がある．本システムは，ローカル・ラグ制御部にて時刻計算の頻度が高く，そのオーバーヘッドが危惧された．そこで本システムでは，独自に実装するタイマをシステム内の時刻情報として用いる．

本システムに実装されるタイマは，システムコールである clock_nanosleep 関数を用いて実現している．具体的には，タイマ専用のスレッドを用意し，その中で clock_nanosleep 関数を用いてスレッドを 0.1ms スリープさせたあと，int 型の数値をカウントアップするという処理を無限ループさせている．これにより，理論上 0.1ms 間隔で動作し，なおかつその値を操作することができるタイマを実現した．このタイマは，同期制御サーバと音響サーバの両方に実装されている．

3.3 3 地点以上におけるローカル・ラグ制御機能

3 地点間の全ての演奏者が同期がとられた演奏音を聞くことができる環境を実現すること

を目指し，それぞれの地点で聴こえる全ての音に同量の遅延を付加することで，各地点で同じ時間に同じ音が聴こえるように制御を行う．具体的には，各地点間の通信遅延の中で最大のものに着目して，その遅延量に合わせて各地点で聴こえる音に遅延制御を行う．

3 地点間以上でローカル・ラグ制御を用いて行う遠隔合奏の原理を図 4 に示す．考え方の基本は，各 2 地点間の遅延量の中で最大の量を全接続地点（全演奏者）に共通の遅延量とするものである．図は 3 地点接続を想定しているが，地点数がさらに増えても図中の手続きはそのまま適用することができる．例えば A 地点に着目すると，A 地点で聴こえる音は，自身のフィードバックと B 地点からのフィードスルー，C 地点からのフィードスルーの 3 種類であるため，これら 3 種類の音が聴こえるタイミングを最大遅延に合わせることを考える．まずフィードバックに関して，本来のフィードバックは遅延が 0ms なので，A 地点でのラグフィードバックのラグ量は最大遅延に合わせて 40ms に設定する．次にフィードスルーに関して，B 地点からのフィードスルーは元々10ms の通信遅延を伴って届くので，最大遅延に合わせるため 30ms 余分に遅延を付加して再生する．この最大遅延に合わせるために余分に付加する遅延を，本稿では補正遅延と呼ぶ．C 地点からのフィードスルーも同様に，20ms の補正遅延を付加して再生する．これにより，自身のフィードバックと B 地点からのフィードスルー，C 地点からのフィードスルーそれぞれが 40ms 遅れて聴こえるという状況が作り出せ，A 地点で聴こえる各音の同期が確保される．これを B 地点，C 地点においても同様に行うことで，全地点で 40ms の遅れは伴うものの，同じ時間に同じ音が聴ける状況を作り出すことができる．

本システムでは，各地点の音響サーバが計測した通信遅延情報を同期制御サーバが集め，最大遅延を計算し，各音響サーバに配信する手順を採用する．各音響サーバが補正遅延を求めるまでの手順を以下に示す．

① 音響サーバ同士が，各音響サーバ間の通信遅延を計測する．計測には，送信側がパケットを送信してから返信を受信するまでの往復遅延時間から，受信側がパケットを受信してから返信を送信するまでの応答までの処理遅延時間を引いて 2 で割るという手法を用いる．

② 各音響サーバが，計測した通信遅延を同期制御サーバに送信する．

③ 同期制御サーバが，収集した遅延情報から最大遅延を求め，各音響サーバに送信する．

④ 各音響サーバは，取得した最大遅延から各地点間の通信遅延を引くことで補正遅延量を求める．ここで用いる通信遅延は，最大遅延を取得する際の手順①で計測した値を用いる．

3.4 音データ処理機構

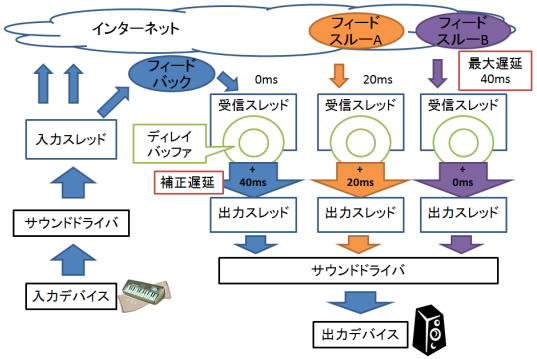

図 5 音響サーバ内の音データフロー．

入力スレッド 入力デバイスから音データを取得し，接続している全ての音響サーバに送信するスレッド．

受信スレッド 対応する地点から送られてきた音データを受信し，受信時刻を記録してディレイバッファに格納するスレッド．音データのチャンネル毎に 1 つ用意されている．

ディレイバッファ 受信時刻が付加された音データを格納するリングバッファ．このバッファから音データを取り出すタイミングを，設定すべき補正遅延量などを用いて調整することで，ローカル・ラグを実現する．

出力スレッド 取り出し時刻を求めてディレイバッファから音データを取り出し，出力デバイスに転送するスレッド．受信スレッド毎に 1 つ用意される．

3 地点接続時において，ある地点における 1 台の音響サーバに着目し，音データの流れと音データ処理機構を図 5 に図解する．まず入力スレッドが音データを入力デバイスから受け取り，自身を含む各音響サーバに送信する．ここでは 3 地点接続を想定しているため，自身を含む 3 つの地点にデータが転送されるが，フィードバックの転送に関してはインターネットを介さずにサーバ内に閉じて行われる．受信スレッドは，音データを受信すると，自サーバの時刻情報を用いて，受信時刻を取得してその音データのヘッダに付加し，ディレイバッファに格納する．出力スレッドは，求めた取り出し時刻に従ってディレイバッファから音データを取り出し，出力デバイスに転送する．取り出し時刻は，自サーバ内の現在時刻からその音に付加する補正遅延時間を引くことで算出する．この時刻が音データに付与された

受信時刻を超えていれば，バッファから音データを取り出す．各出力スレッド毎に取り出した音データはサウンドドライバでミキシングされ，1 チャンネルの音として出力デバイスに転送される．

3.5 時刻同期機能

3.2 節に紹介したタイマを用いて，時刻同期機能を実現する．具体的には，同期制御サーバが自身のタイマのカウント数を各音響サーバに配信し，その値に従って各音響サーバが自身のタイマのカウント数を補正することで，各音響サーバのタイマの同期を確保する．全ての音響サーバのタイマが同期するまでの手順を以下に示す．

① 接続を予定していた全ての音響サーバが接続を済ませたことを確認し，時刻同期処理が開始される．

② 同期制御サーバが，各音響サーバとの通信遅延を計測する．

③ 同期制御サーバが，リセット値として現在のタイマのカウント数に通信遅延をタイマ換算した値を付加し，各音響サーバに送信する．この時付加する通信遅延は，それぞれの地点に応じた値を用いる．

④ 各音響サーバは，同期制御サーバから取得したリセット値を自身のタイマの値に反映させる．

以上の手順で各サーバのタイマが同期される．タイマ同期後，同期制御サーバが指定時刻に録音ファイルをオープンするという命令を各音響サーバに送信すれば，共通のタイムスタンプを持つ録音ファイルが作成できる．

3.6 実　　装

本システムを構成する音響サーバと同期制御サーバの双方を実装するコンピュータの基本的諸元を以下に示す．

- OS:Fedora14
- カーネル：2.6.35.6
- サウンドドライバ：ALSA（I/O ライブラリである PortAudio を介して使用）
- 音データのフレーム長：約 23.2ms
- サンプリングレート：44.1kHz
- 量子化ビット数：16bit
- チャンネル数：1（モノラル）

本システムが独自に実装する独自タイマには，Linux2.6.21 以降からサポートされている高精度タイマ（High-Resolution-Timers）を用いている．さらに，独自タイマの精度を向上させるため，タイマ機能を実現するスレッドに対してリアルタイムスケジューリングを適用する．スケジューリングポリシーは SCHED_FIFO を採用し，タイマスレッドを最優先で処理させるように設定する．なお，このリアルタイムスケジューリングを用いない場合のタイマ精度が低下することは，実験によって確認済みである．また，実装コンピュータのハードウェア諸元は 4 節に紹介する．

4. 性能評価実験

4.1 タイマの精度計測

4.1.1 実験環境と実験手順

タイマの周期が，指定した周期に対してどの程度ずれているか，また，その周期がカウントアップごとにどの程度ずれているかを計測し，タイマの周期精度を明らかにする．ここで，タイマの周期とは，例えばタイマのカウント数が 0 から 1 にカウントアップされるまでの時間間隔のことを指す．

実験では，同じ仕様の CPU を持つ 2 台のコンピュータ（以下マシンと呼ぶ）と，その 2 台より CPU の性能が劣るマシン 1 台の計 3 台を使用した．ほぼ同じ CPU を持つ 2 台のマシンをそれぞれマシン A，マシン B とし，マシン A，B よりも CPU の性能が劣るマシンをマシン C とする．以下に，各マシンの諸元を示す．

マシン A，B

- CPU：Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz

マシン C

- CPU：Intel(R) Core(TM)2 CPU 6400 2.13GHz

実験の手順を以下に示す．

① タイマを実現する clock_nanosleep 関数のスリープ時間を 0.1ms に設定して，関数を呼び出してから処理が返ってくるまでの時間を計測する．タイマのカウント数が 100,000 になるまでの計測を 1 回の試行とし，これを 5 回行う．

② 1 回の試行で得られた 100,000 個の周期に関する平均と標準偏差を算出する．

4.2 実験結果と考察

計測して得られた平均を図 6 に，標準偏差を図 7 に示す．また，マシン A の，ある 1 回の試行における 50,000〜51,000 カウントまでのタイマ間周期をグラフ化したものを図 8 に示す．

図 6 と図 7 より，平均で見ればそれぞれの誤差は非常に小さく，安定して動作しているよ

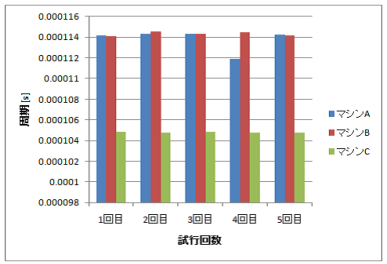

図 6 タイマ周期の平均.

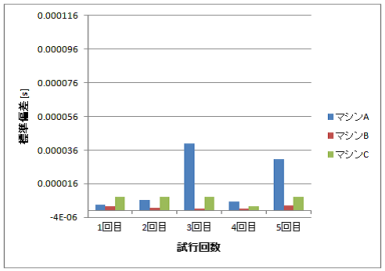

図 7 タイマ周期の標準偏差.

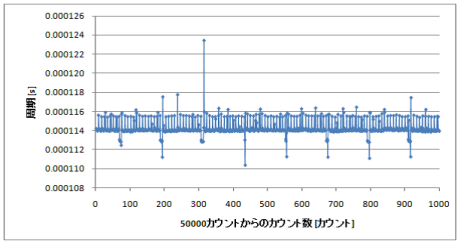

図 8 マシン A のある試行における 50,000〜51,000 カウントまでのタイマ周期.

うに見える．タイマ同期の平均値における誤差は，0.01ms 程度であり，標準偏差は 0.04ms 程度である．しかし，図 8 から，実際の周期の変動は多くの場合は安定しつつも，周期的に大きくずれるということを繰り返す特徴的な性質を持っていることがわかる．

4.3 時刻同期機能の精度計測

時刻同期機能を用いてタイマを同期させた直後の各音響サーバのタイマのカウント数の誤差を計測することで，時刻同期機能の精度を明らかにする．

4.3.1 実験環境と実験手順

計測のために構築した実験環境を図 9 に示す．4.1 節の実験で使用したマシンを用いて実験環境を構築した．マシン A とマシン B 上で音響サーバを動作させ，マシン C 上で同期制御サーバを動作させる．本実験においては，マシン A を音響サーバ A，マシン B を音響サーバ B，マシン C を同期制御サーバを呼ぶ．各サーバは同一の LAN に接続し，通信遅延は Dummynet と呼ばれる通信制御ツールを用いて発生させた．また，同一の電子メトロノームからの出力を二股に分けて，各音響サーバの音入力口に接続している．電子メトロノームから出力される音源は，連続的に音が鳴り続けるビープ音を用いる．各音響サーバには，実験用に，無音ではない何らかの音が入力された瞬間にタイマのカウント数を出力するような実装を施した．具体的には，入力された音データが格納されている配列の要素値を参照し，ある一定の閾値を超えている場合は音が入力されたと判断するようにしている．

実験の手順を以下に示す．

① 音響サーバ A，B の順に時刻同期機能を用いてタイマの同期を行う．
② 各音響サーバのタイマが同期したことを確認し，電子メトロノームから音を入力する．各音響サーバのタイマのカウント数がそれぞれ出力される．
③ ②で得られたカウント数の誤差を計測する．誤差は音響サーバ A のタイマのカウント数から音響サーバ B のタイマのカウント数を引くことで算出する．
④ 1 つの誤差を求めることを 1 回の試行とし，通信遅延を 0ms，10ms，20ms，30ms，40ms，50ms と 6 段階に変化させて行う．
⑤ 通信遅延を変化させた 6 回の試行を 1 セットとし，3 セット行う．

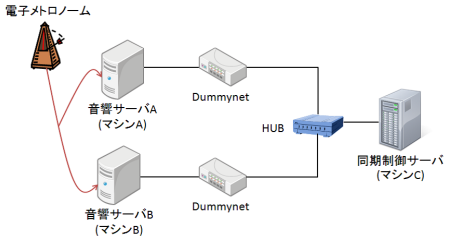

図 9 時刻同期精度の実験環境.

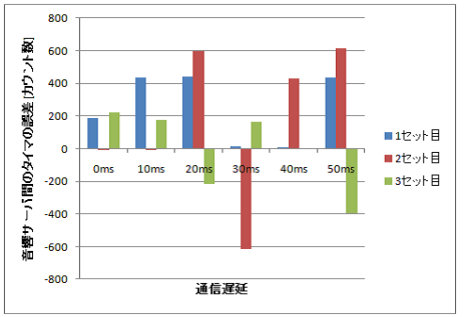

図 10 時刻同期後の各音響サーバ間のタイマ誤差.

4.3.2 実験結果と考察

実験によって得られた結果を図 10 に示す．図 10 の，正方向へのカウントが多い場合は，音響サーバ A のタイマの進みが早いということを示す．電子メトロノームからの入力のタイミングは手動であるが，その間は 1〜2 秒程度であった．タイマ周期の誤差が影響する時間をタイマ換算すると 10,000〜20,000 カウントとなり，その間に各音響サーバ間のタイマ周期が図 8 のように不規則に変動すれば，この程度のずれは生じ得る．図 10 中の結果はこの不規則さを反映したものと思われる．

4.4 ローカル・ラグ制御精度

2 地点間接続にローカル・ラグを適用した際の，ローカル側で再生されるラグフィードバックとリモート側で再生されるフィードスルーのずれを計測し，ローカル・ラグの制御精度を

明らかにする．実験では，ローカル・ラグ制御に用いる時刻として，独自タイマを用いる場合と，システムクロックを用いる場合の 2 通りで行い，それぞれの制御精度について比較を行う．これは，前述の独自タイマの評価結果を受け，もう一つの選択肢であるシステムクロックを用いる同期制御の性質を調べる必要もあると判断したためである．なお，実験環境や実験手順に関してはどちらも同一である．

4.4.1 実験環境と実験手順

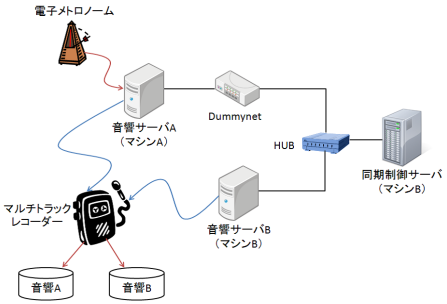

図 11 ローカル・ラグ制御精度の環境

構築した実験環境を図 11 に示す．マシンとサーバの対応は前節と同様である．全てのサーバを同一の LAN に接続し，Dummynet を音響サーバ A と HUB の間にのみ配置する．120BPM の電子メトロノーム音を音源とし，音響サーバ A のみに入力する．各音響サーバからの出力をマルチトラックレコーダーに接続し，同一タイミングで録音を行う．実際に演奏者が聴く波形に関して同期がとれているかどうかを調査するためにこの方法を採用した．

構築した環境でシステムを動作させ，通信遅延を 0ms から 100ms まで 20ms 刻みで 6 回計測する．1 回の計測は 10 分程度行う．1 回の計測で得られた 2 つの録音ファイルの，0 分（ファイルの 1 音目），2 分，4 分，6 分，8 分，10 分それぞれの時間で誤差を計測した．

4.4.2 実験結果と考察

システムクロックを用いた場合の計測結果を図 12 に，タイマを用いた場合の計測結果図 13 に示す．

図 12 と図 13 を比較すると，タイマを用いた制御に比べ，システムクロックを用いた制御の方が安定して動作している．このことから，ローカル・ラグ制御にはシステムクロックを用いたほうが，より高精度な実験が行えるということがわかった．

システムクロックの誤差に関して，約 20ms ほどの誤差が生じる個所も多々あった．これ

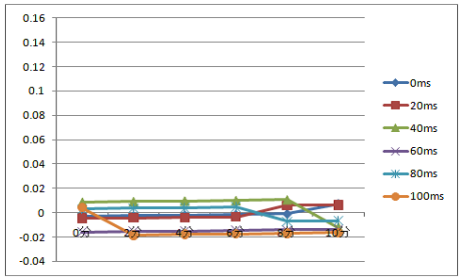

図 12 システムクロックを用いたローカル・ラグ制御精度

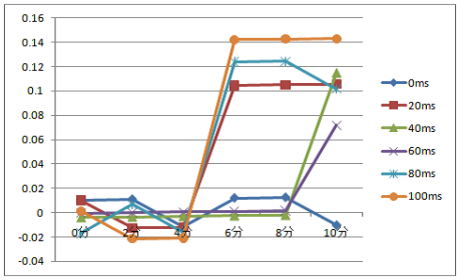

図 13 タイマを用いたローカル・ラグ制御精度

は，本システムで扱っている音データのフレーム長が約 23.2ms であることに起因すると考えられる．フレーム長分の周期で 1 つの音データをデバイスから取得するため，受信スレッドで付与される受信時刻が 23ms 周期でしか取得出来ない．このことが，ローカル・ラグ制御に影響を及ぼしているのではないかと推測する．

5. ま と め

ローカル・ラグの評価実験に特化した基盤システムの構築を目指し，3 地点以上の接続へのローカル・ラグ適用機能と，時刻同期機能の遠隔合奏支援システムを開発した．開発したシステムの性能評価実験を行った．その結果，ローカル・ラグ制御にはシステムクロックを用いることで，より高精度な実験が行えることがわかった．

謝辞 本研究を進めるに当たり，システムの設計から開発を通して貴重なご意見を賜りました山口毅氏に心よりお礼申し上げます．

参 考 文 献

1) Dane Stuckel and Carl Gutwin; The Effects of locallag on Tightly-Coupled Interaction in Distributed Groupware : Computer Supported Cooperative Work, pp.447-456(2008).
2) 入江洋介, 青柳滋己, 高田敏弘, 平田圭二, 梶克彦, 片桐滋, 大崎美穂; t-Room のための遠隔合奏支援システムの構築, 情報処理学会, Vol. 2009-GN-73, No. 3 (2009. 11).
3) Juan-Pablo C ´ aceres, Chris Chafe(Center for Computer Research in Music and Acoustics (CCRMA)); JACKTRIP/SOUNDWIRE MEETS SERVER FARM: Proceedings of the SMC 2009 - 6th Sound and Music Computing Conference, 23-25 July 2009 (2009)
4) NETDUETTO — Y2PROJECT; http://www.y2lab.com/project/netduetto/