卒業論文　　2003年度 (平成15年度)

分散環境におけるフィードバックを用いたオーケストラ演奏機構の構築

慶應義塾大学 環境情報学部

氏名：久松 剛

指導教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

南 政樹

平成15年12月29日

卒業論文要旨 2003 年度 (平成 15 年度)

分散環境におけるフィードバックを用いたオーケストラ演奏機構の構築

本研究では，ネットワークを用いたオーケストラ演奏機構の構築を行った．

ネットワークを用いた共同制作，特にリアルタイムで行われるオーケストラ音楽の共同制作はフィードバックによる他の演奏者のモニタリングによる和音，ダイナミックスの確認，タイミングデータによる演奏のタイミングやリズムの共有，演奏データの時間軸取得，演奏者ごとに異なるネットワーク伝送遅延時間の管理の 4 つを行う必要がある．

本研究ではネットワークを介した分散環境における音楽共同制作モデルを提案し，モデルに基づいた設計と実装を行った．

本研究では提案したモデルに基づき 5 つのモジュールを作成した．クライアント管理部ではクライアント毎に遅延時間や IP アドレス，映像・音声データ送信用ポート番号といった情報を管理する．遅延管理部では各データの時間軸管理，ネットワーク伝送遅延の管理を行う．管理されたデータを基にした同期は本論文では対象としない．フィードバック送受信部は他の演奏者の音声データの送受信を行う．タイミングデータ送受信部は演奏のタイミング，リズムの演奏者間共有を行う．映像・音声データ送受信部では演奏者が演奏した演奏を映像と音声の形式で転送を行う．これらのモジュールを DVTS for MacOSX の機能拡張として実装，評価を行った．

本研究により，ネットワークを介した音楽共同制作が可能となった．映像・音声データを対象として扱うため，放送中継など他分野のリアルタイムコンテンツの共同制作にも応用可能である．

キーワード
1. 実時間配信，2. マルチメディア，3. 分散環境, 4. インターネット

慶應義塾大学 環境情報学部

久松 剛

Abstract of Bachelor's Thesis

Academic Year 2003

Establishment of orchestral performances mechanism with feedback in distributed environment

In this research, a mechanism for supporting realtime co-operative orchestral performance between remote places with the use of network is designed and implemented.

Supporting realtime co-operative orchestral performance needs to consider next four matters through feedback from other performers. They are, confirmation of harmony and dynamics by monitoring co-performers, sharing rythm and timing of the performance using timing data, aquisition of time stamps on music data, and management of network time delay between each players' location.

This research proposes a model for producing music co-operatively in distributed environment with the use of networking technologies. A system based on proposed model is designed and developped to show the advantages of this research.

The system consists of five modules. They are, client manager, delay manager, feedback data transmitter/receiver, timing data transmitter/receiver, and video and audio data transmitter/receiver. Client manager module manages the information such as IP address and port numbers as well as network delay time between each clients. Feedback data transmitter/receiver transmits the performer's audio data and receives co-performer's audio data. Timing data transmitter/receiver rythm and timing of the performance between performers. Video and audio data transmitter/receiver transfers performance created by this system to viewers. However, automatic synchronization of the performance using timing data is not considered in this research. Above modules are implemented as extensions for DVTS for MacOSX and evaluation is taken using this implementation.

In conclusion, realtime co-operative orchestral performance with the use of networking technology is achieved. This system may be applied for producing other realtime contents such as live telecasts since it is designed for use with video and audio data.

Keywords :
1. real time transport, 2. multimedia, 3. distributed environment 4. internet

Keio University , Faculty of Environmental Information

Tsuyoshi Hisamatsu

目 次

図 目 次

表 目 次

第1章　はじめに：分散環境における音楽共同制作環境に向けて

1.1　概要：ネットワークを用いた共同制作

IEEE802.3ab[1], IEEE802.3ae[2] といったネットワークの広帯域化に伴い，データサイズの大きな品質の高い映像・音声データを，リアルタイムに双方向，複数転送することが容易となった．

これにより，ネットワーク上でのマルチメディアコンテンツ共同制作活動が活発化している．例えばデジタルシネマプロジェクト [3] では，遠隔のスタジオ間での撮影作業が試みられた．インターネットを用いた制作作業には，共同制作を行う際の地理的な制約がなくなり，制作に関するコストが下がるという利点がある．映像制作だけでなく，音楽制作，放送中継などにもネットワーク上の共同制作は有効である．

デジタルシネマプロジェクトのような映画制作では，収録後に CG などによる加工，合成，編集，吹き替えなどの作業によって修正・編集が加えられた後，視聴者へとコンテンツが提供される．しかしライブコンサート，放送中継といったリアルタイムに視聴者に配信される分野では，実時間性が求められるため，コンテンツの加工や修正は困難である．

1.2　本研究の目的：フィードバックを有する共同制作環境の構築

本研究の目的は，広帯域化・低遅延化が進む IP ネットワーク環境を利用したマルチメディアコンテンツのリアルタイム共同制作環境を実現することである．本論文ではリアルタイム共同制作環境の実現に向け，以下の 3 つに着目し，分散環境におけるリアルタイムメディア共同制作を実現する．

1. 共同制作者間の連携
   他者の音声をモニタリングすることによる分散環境における共同制作の効率化
2. 共同制作者間の時間軸共有
   動作の開始，間隔を制作者間で共有
3. 拠点毎に異なる遅延時間ネットワーク伝送遅延，収録環境・転送機器の差異による遅延時間の取得

リアルタイムメディア共同制作のうち，上記の 3 点の要求が厳しいものの 1 つにオーケストラのライブコンサート，練習がある．本研究ではリアルタイムメディア共同制作の一つとしてオーケストラを対象に設計，実装を行う．

1.3 論文の構成

本論文は 8章から構成される．

第 2章では実空間における音楽共同制作の特徴を，特にオーケストラを対象として述べた後，分散環境上で実現する際の要点，本論文で提案する音楽共同制作モデルについて述べる．

第 3章では本研究における要素技術として複数ストリームの転送を行うための要素技術に関して述べる．

第 4章において，オーケストラを対象とした要素技術について述べる．インターネットを用い音楽の共同制作の関連研究として坂本龍一 MIDI ライブ，長野オリンピック開会式の事例を述べた後，ネットワークを用いて音楽を製作するシステムとして Yamaha 社の mLAN に関して述べ，それぞれの問題点をあげる．

第 5章においてその問題を解決することのできる，本システムの設計に関して述べる．

第 6章で設計に基づいた実装に関して述べる．

第 7章で本システムの実装に対する評価を行い，既存の問題を解決したか否かを述べる．

最後に第 8で本研究のまとめ，及び今後の展望を述べる．

第2章　提案：分散環境における音楽共同制作

本研究ではネットワーク上のマルチメディアコンテンツ共同制作の対象として，音楽の共同制作，特にオーケストラに焦点を当てる．

本章では，コンサートホールに演奏者が集合する実空間における音楽共同制作の特徴について述べた後，本研究での音楽共同制作モデルの提案を行う．

2.1　実空間における音楽共同制作

オーケストラのコンサートは通常，コンサートホールにて行われる．コンサートホールにおける演奏環境を図 2.1に示す．ステージ上の演奏者は指揮者の指揮に従い，演奏を行う．演奏された音声はステージ後方と左右に設置された反響版，ホール天井，壁などにより反響する．視聴者はステージ前方の客席に位置し，楽器からの直接音と反響音を聴取する．ステージ上では距離によっては音声が遅れて到達するが，視聴者には同期が確保された状態で届く．これはコンサートホールの設計によって反響音を操作し，同期を確保しているためである．

コンサートホールにおける合奏では，壁や天井からの反響時間が長く，ステージに音が返って来る際には遅延が発生する．特にステージからの距離が離れている演奏者の音声は遅延して到達するため，聴覚による演奏のタイミングの取得は困難である．聴覚ではオーケストラ全体としての和音，ダイナミックスの確認がなされる．このような演奏形態を実空間における音楽共同制作と定義する．

演奏者，指揮者，視聴者がネットワーク上に分散した環境において図 2.1の環境を実現する場合の，共同制作環境の要件として以下の 3 つが挙げられる．

- 演奏者が他の演奏者の音声聴取
- 演奏者間のタイミング，リズム共有
- 視聴者は全ての演奏者の演奏を同期された状態で聴取

2.2　分散環境における音楽共同制作

3 つの要件を元に，本論文で想定する分散環境における音楽共同制作環境のモデルを図 2.2に示す．中心を指揮者のいる拠点 D とする．まず指揮者の映像を各拠点の演奏者に送信する．タイミングを取得するための指揮者映像をタイミングデータと呼ぶ．送信された映像を元に各演奏者は演奏を行い，拠点 D へと演奏データを送信する．拠点 D は各拠点から受信した演奏データを各拠点に送信する．これにより，各拠点の演奏者は他の演奏者の音を聴取することができる．他の演奏者の音声データをフィードバックデータと呼ぶ．

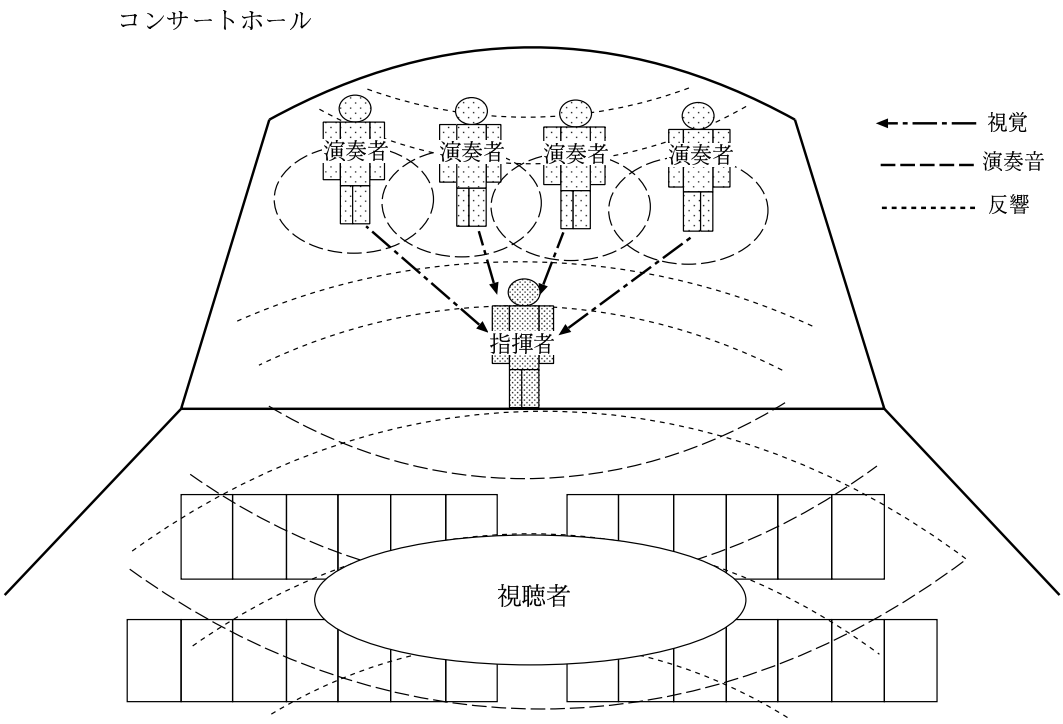


図 2.1: 実空間における演奏

各拠点から受信した演奏データは，バッファ，ソフトウェアミキサなどと連携し，音声レベル合わせなどが行われる．視聴者に対してはミキサから出力された演奏データをインターネットなどを経由して各家庭などに届けられる．

フィードバックデータは，タイミングデータを元に演奏を行った演奏者が他の演奏者の音声を受信するまでの時間が，演奏に影響を与えない範囲で演奏者が聴取する必要がある．このため演奏の基準となるタイミングデータは演奏可能時間に影響を与えない範囲での送信が必要である．演奏されたデータのミキシング，出力はバッファリングによる同期処理を経て行われる必要がある．バッファリング，ミキシング時間が必要なため，視聴者に対する音声データの送信には実時間性を求めない．また，本研究では視聴者に対する出力の実時間性が実現された上で可能な視聴者からのフィードバックは対象外とする．

これらのデータは全て安定したデータ転送がなされる必要がある．特に映像・音声データの送信が不安定な場合，音声が途切れるなどの影響があるため，合奏全体の品質に影響する．

途中経路に遅延発生装置を使用，ネットワークを介した異なる拠点の二人の演奏者に音のみの到達性を確保し，合奏を行ったレポート [4] によると，許容遅延時間範囲は 50ms-80ms とされている．この許容値より，異なる拠点のクライアントを $P_{(1)}\cdots P_{(2)}$ とした場合，それぞれのクライアントに対し，タイミングデータ出力時間からフィードバック受信時間の差分 $Pd_1\cdots Pd_n$ が存在する．これより，演奏可能許容遅延時間は

$$Max(\overline{Pd_{(1)-(n)}}) = Delay < 80ms$$

によって求められる．

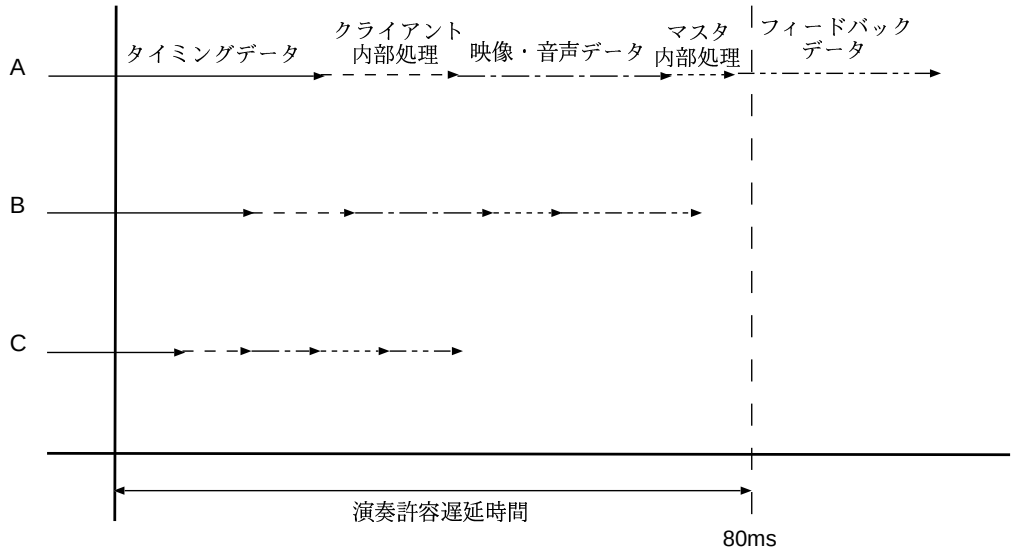


図 2.2: 分散環境における音楽共同制作モデル

本機構におけるタイミングデータ，フィードバックデータと演奏可能許容遅延時間の関係を図 2.3に示す.

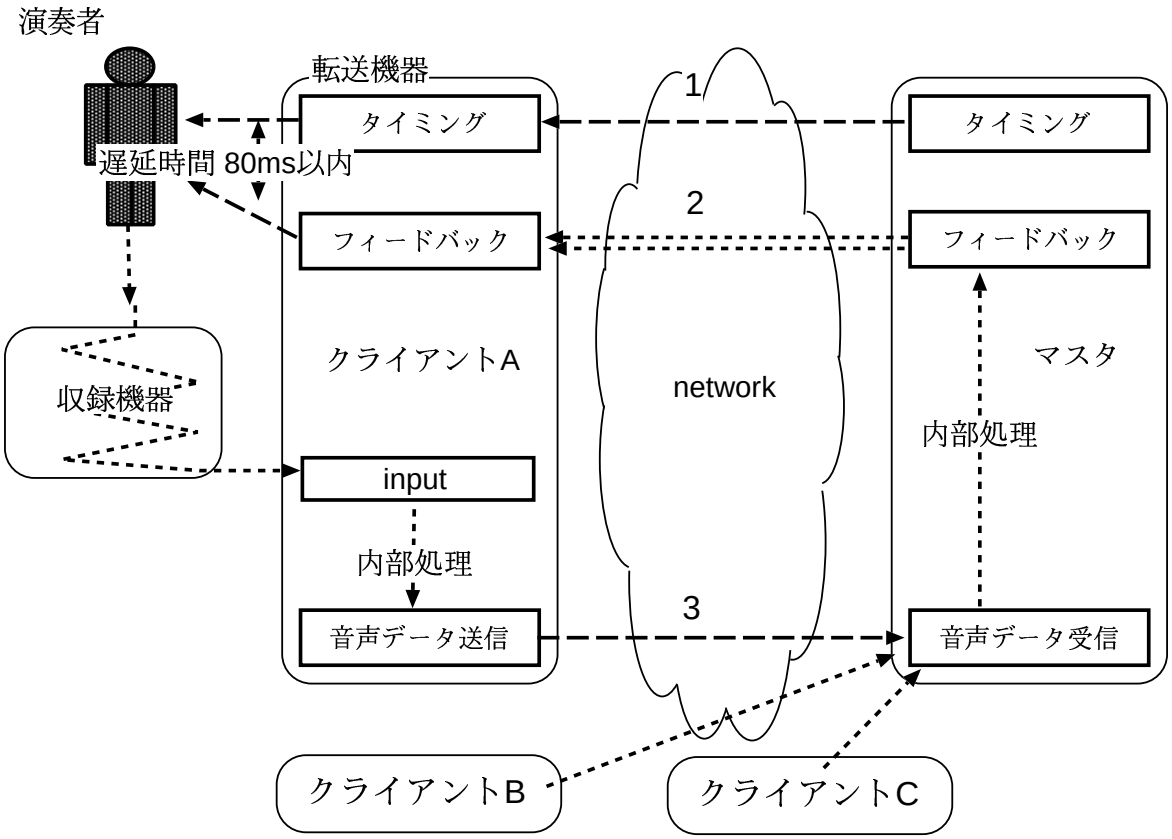


図 2.3: 許容遅延範囲 概念図

2.3　まとめ：分散環境における音楽共同制作モデルの提案

本章ではまず実空間における音楽共同制作の分析を行った．これにより，分散環境における音楽共同制作に必要な要件として他の演奏者の音声聴取，演奏者間のタイミング，リズム共有，視聴者に対し，全ての演奏者の音声が同期された状態での出力の 3 つがある．次に要件を実現するためのモデル提案を行った．次に演奏可能許容遅延時間を数式で示した．次章では本章で提案したモデルに基づいた機構構築のために必要となる基礎技術について述べる．

第3章　音楽共同制作環境の構築に必要な基礎技術

本章では，本機構を構築するために必要である IP ネットワークを用いた映像・音声といったマルチメディア転送，及び複数ストリームの同期に用いられる技術に関して述べる．

3.1　IP ネットワーク上における映像・音声転送

本節では IP ネットワーク上における映像・音声転送に用いられる，ストリーミング技術に関して述べる．

3.1.1　ストリーミング

映像・音声データ全体を受信してからローカルで再生する方式に対し，データの受信と再生を並行して行う方法がストリーミング再生である．リアルタイム性の高い映像・音声転送だけでなく，Real Systems[5] に代表される蓄積型映像・音声配信にも用いられる．

IP ネットワークを用いた転送では，途中経路における伝送メディアの速度，ネットワークの帯域，ネットワークの混雑状況によりネットワーク伝送時の遅延 (ネットワーク伝送遅延) が生じる．遅延時間は，途中経路の変更などにより揺らぎが発生する．そのため，映像・音声の再生に対してデータの受信が遅れる場合が発生し，映像・音声が正しく再生されない．特に音声は受信と再生のタイミングが異なる場合には再生が途切れることがある．

ネットワーク伝送遅延を吸収するために，受信したデータを一時的にバッファに蓄積，蓄積されたデータから再生を行うバッファリングと呼ばれる方法がある．バッファサイズが大きいほど遅延時間の揺らぎを吸収することができるが，データ再生が遅れるため，再生遅延時間も大きくなる．

複数拠点からのストリーミングを行い，一箇所で受信する場合，ネットワーク伝送遅延が拠点によって異なるため，拠点ごとに遅延時間，バッファリングの管理を行う必要がある．

次節では，これらのネットワークを利用したストリーミングに用いられるプロトコルに関して解説する．

3.1.2　転送プロトコル

実時間通信を行う映像･音声転送では2つの特徴により，TCP(Transmission Control Protocol)[6] ではなく，UDP(User Datagrame Protocol)[7] が有利な場合が多い．

- TCP による再送制御が発生するとパケットの到着時間に遅れが生じる．実時間再生を行う受信側に対し，TCP による再送制御を行った場合，再送時間に対するパケットの到達

時間が間に合わず，映像や音声が乱れる可能性がある．UDP を用いた通信は，TCP を用いた通信と比べ，トランスポートレイヤにおけるパケットの到達順序が異なる可能性がある．パケット到達順序の判別を行うにはアプリケーションでの対応が必要となる．到達順序の判別を行うための代表的な手法として RTP[8] がある．

- TCP による輻輳制御が発生するとパケットの到着時間に影響が生じる．輻輳制御が行われた場合，受信側で実時間に基づいた再生に対して間に合わず，映像や音声が乱れる可能性がある．そのため RTP と RTCP(Real Time Transport Control Protocol)[8] を併用することでパケットロスを検知し，映像レート，フレーム数などを変化させることにより，転送量の調整を行うなどの対応がなされている．

RTP は end-to-end のネットワークにおいて映像・音声といった実時間性が必要とされるデータ転送に利用する目的のために，トランスポートレイヤとは独立して設計されているプロトコルである．RTP フォーマットを図 3.1に示す．送信時に RTP によって Sequence Number の付加による送出パケットの順番を示し，Time Stamp に送出時の時間データを組み込むことで時間軸を示す．これにより受信側においてパケットの順序確認を行う．

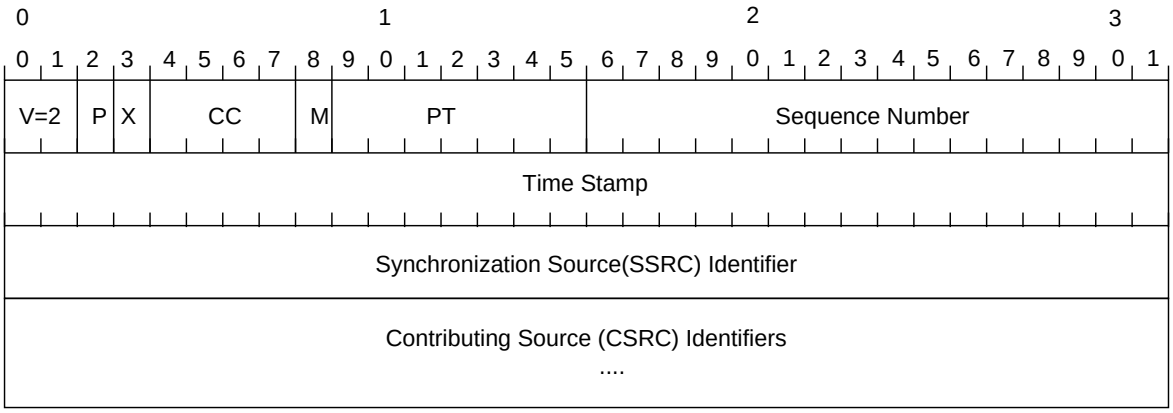


図 3.1: RTP フォーマット

3.2 まとめ：分散環境におけるメディア共同制作の問題点

本章では本モデルに基づいた機構構築のために必要な基礎技術について述べた．次章ではネットワークを用いる音楽の共同制作機構の紹介を行い，本モデルとの比較，検証を行う．

第4章　既存技術: インターネットを用いた音楽の共同制作

本章では既存のネットワークを用いた音楽制作技術について述べる．また第 2章で述べた本研究で提案するモデルとの比較，技術的問題点と合わせた検証を行う．

インターネットを用いた音楽の共同制作の事例として坂本龍一 MIDI ライブ [9]，長野オリンピック開会式 [10] がある．IEEE1394[11] を用いたネットワーク上で音楽の共同制作を行うものに Yamaha 社の mLAN[12][13][14] がある．

4.1　坂本龍一 MIDI ライブ

坂本龍一 MIDI ライブは，インターネットと MIDI を用いたライブ中継である．コンサート会場で行われた演奏は国内のヤマハ支店，個人宅に向けて送信される．インターネットと MIDI を用いたライブ中継イベントであり，1997 年 1 月から 12 月まで 3 回にかけて行われた．

4.1.1　坂本龍一 MIDI ライブの特徴

システム構成図を図 4.1に示す．システム構成はそれぞれの回に共通する．視聴者に対しピアノ演奏の音声データ，オーケストラや肉声などの音声データ，コンサートホールの映像はそれぞれ独立したストリームで転送される．

ピアノには MIDI[15] 対応のものが用いられ，PC を経由してインターネットに MIDI ストリームを MidLive[16] を用いて送信する．オーケストラ，肉声などは TwinVQ(Transform-domain Weighted interleave Vector Quantization)[17] をヤマハ社が拡張した SoundVQ[18] に変換された後，中継サーバを経由して視聴者に送信される．コンサートホールの映像は RealServer[5] にて Real 形式に変更され，視聴者に送信される．

クライアントは，送信されたそれぞれのストリームをインターネット MIDI ライブシステム [19] を用いて受信し MIDI 対応ピアノ，及びオーディオ機器へと出力する．インターネット MIDI ライブシステムではクライアント PC で専用ソフトを用いたバッファリングを行い，それぞれのストリームの同期を取った上で視聴者に届けられる．

4.1.2　坂本龍一 MIDI ライブの検証

Midi ライブシステムは複数のストリームを扱う点，それぞれのストリームは同期が取られた上で視聴者に届くという点で本研究の対象とするモデルと類似している．

Midi ライブシステムは演奏者対視聴者のサーバ・クライアント型の片方向通信である．演奏者はコンサートホールに集合，実空間において演奏を行うため，本研究の対象とするモデルと

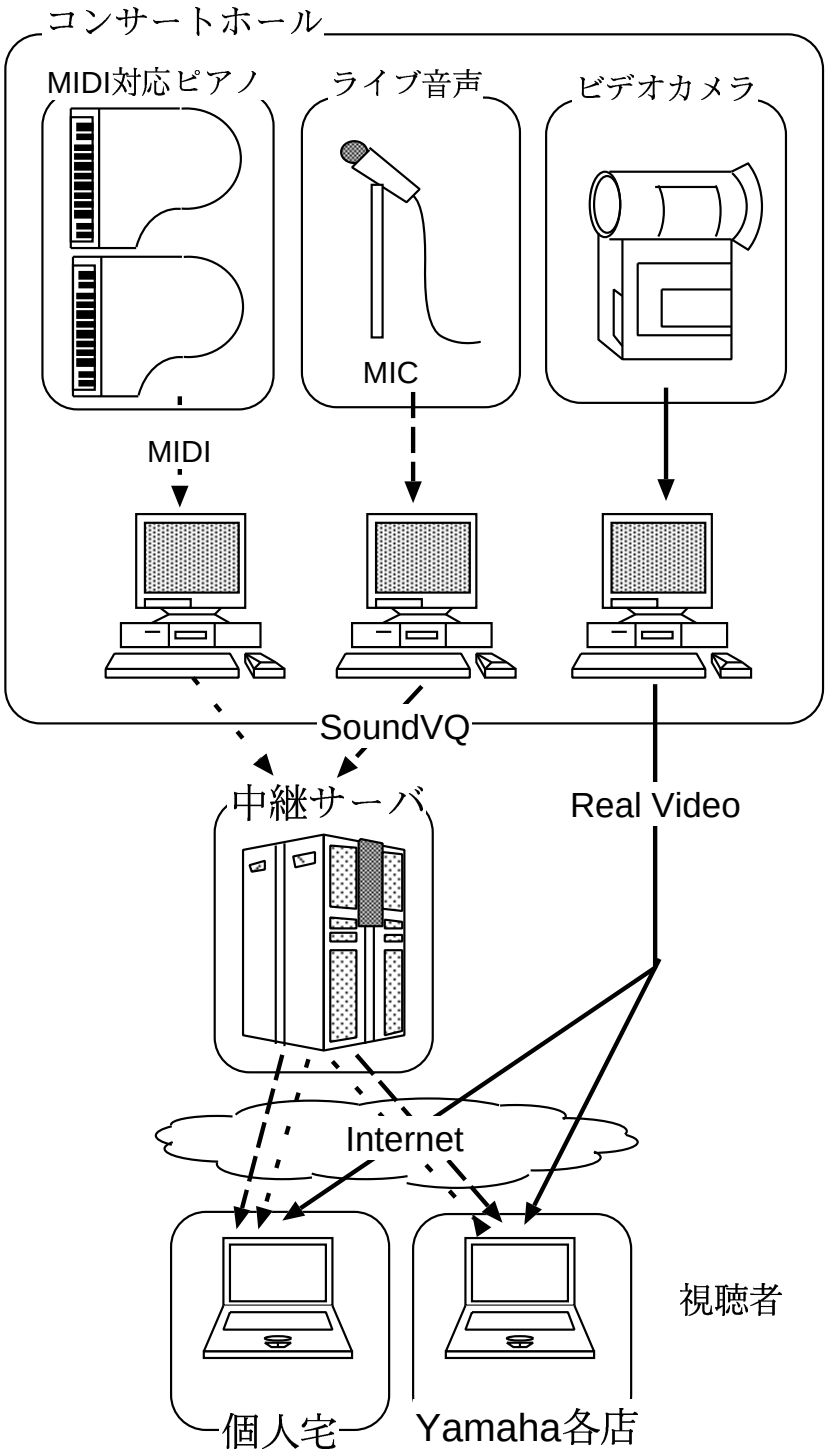


図 4.1: 坂本龍一 MIDI ライブ システム構成

は異なる.

4.2 Yamaha mLAN

4.3 yamaha-mlan

MIDI 対応デバイスなどを相互に接続し，音楽の共同制作を行うシステムとして Yamaha 社の mLAN[12] がある．本論文では mLAN をインターネットではないネットワークを用い，厳密な時間軸管理を行う機構の例として取り上げる．

mLAN はシンセサイザーやデジタルミキサーなどの電子楽器，機器，コンピュータなどを IEEE1394 を用いて接続し，MIDI データやオーディオデータを送受信を行うシステムである．システム構築例を図 4.2に示す．IEEE1394 に対応した MIDI デバイス，光ケーブル等を用いたネットワークを構築することができる．接続された機器の音声データは，IEEE1394 ケーブルを介して PC などに出力することができる．

mLAN で扱えるサンプリングレートは 32kHz, 44.1kHz, 48kHz, 96kHz である．ビットレートは 24bit，32bit, 64bit の他，raw format である．mLAN 対応デバイス，PC の他，既存のオーディオ機器や MIDI 機器を mLAN 製品のアナログ端子，Optical 端子に接続することにより既存の楽器，収録環境を生かしての利用が可能となっている．

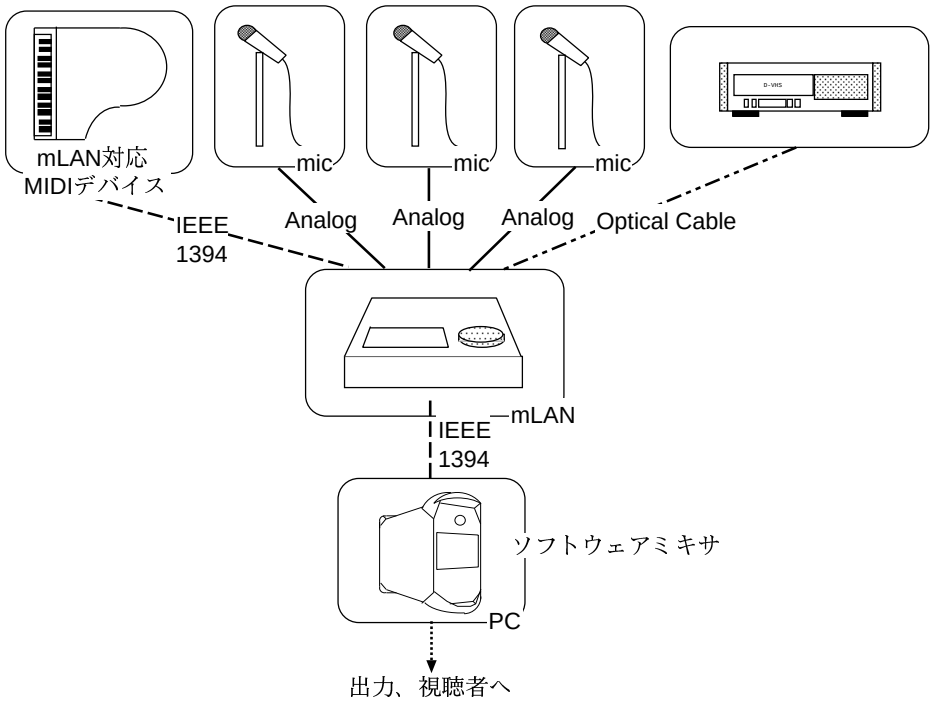


図 4.2: mLAN システム構成例

4.3.1 IEEE1394 を用いた音声データ転送

本稿では，mLAN で採用されているインターフェースである IEEE1394 に関して述べる．

IEEE1394 は 1995 年に”IEEE Standard for a High Performance Serial Bus”[11] として発表された高速シリアルバスインターフェースである．IEEE1394 は以下の特徴を持つ．

- 速度
  既存の規格である IEEE1394-1995[20],IEEE1394a-2000[21] では 100Mbps, 200Mbps, 400Mbps の転送速度が実現されていた．2002 年に規格化された IEEE1394b-2002[22] の beta モードでは 800Mbps, 1.6Gbps，3.2Gbps といった転送速度が実現される．

- 接続距離
  既存の規格である IEEE1394-1995 では，最大長は 4.5m であった．IEEE1394b-2002 では伝送媒体に CAT-5UTP，MMF を用いることで 100m まで距離を伸ばすことができる．

- Isochronous 転送，Asynchronous 転送
  Isochronous 転送は 125 μ 秒ごとに必ずデータ転送ができることを保証する転送機能である．このため一定時間内に一定量のデータ送信が必要な画像・音声転送に利用されることが多い．mLAN は Isochronous 転送を用いて音声データの読み出し，書き込み，転送を行う．
  Asynchronous 転送は受信側がデータ到着に対する応答を返す．遅延時間は保証されないが，確実なデータ転送が可能なため，通常のデータ転送に用いられることが多い．

4.3.2 mLAN における機器間同期機構

複数の機器を用いての音楽の共同作業を行う際に問題となることに，機器間の同期がある．mLAN における同期機構を図 4.4に示す．デジタル音声データは 1/Samplig_Frequency で定間隔を保った状態で扱われ，それぞれのデータにはタイムスタンプが押される．ネットワーク上では図 4.3のように 1/Transfer_Frequency の間隔で複数のデータが纏まった状態で扱われる．音声データに復元する際，音声データに含まれるタイムスタンプを基に復元するが，パケット変換時にタイムスタンプを管理するチップセットの時間軸のずれ，及びネットワーク転送時の遅延時間，遅延の揺らぎが問題となる．mLAN ではデータのやりとりを行うノードの他に，第三者として Clock Master と呼ばれるノードが，他のノードに対してシリアルバスクロック修正を行う機構を設けている．これにより各クライアントのチップセットの同期を行っている [13].

本研究では伝送メディアとして長距離伝送が可能なインターネットを対象とするが，インターネットでは伝送遅延時間の揺らぎなどが大きい．そのため音声データの時間軸管理，本論文では対象外とするが転送機器の内部処理の管理を行う必要がある．

4.3.3 mLAN の検証

mLAN では MIDI ベースで音声データのやりとりがなされるため，時間軸管理が正確であるという利点がある．機器内チップセットが原因となるクロックの差異もまた第 4.3.2項で述べた Clock Master の利用により，修正される．

mLAN は IEEE1394 を用いたローカルネットワークを前提としたものである．そのため最長距離は IEEE1394b[22] の規格である 100m が限界となることから都市間などの長距離での利用は不可能である．長距離利用を手軽に行うためにインターネットを利用する場合，拠点毎に異なる遅延時間，遅延の揺らぎを考慮した音声データの時間軸管理，ミキシングを行うための同期処理を行う機構が必要となる．

また，mLAN では映像の利用は想定外であるため，各拠点の演奏者の映像を見る場合は全く

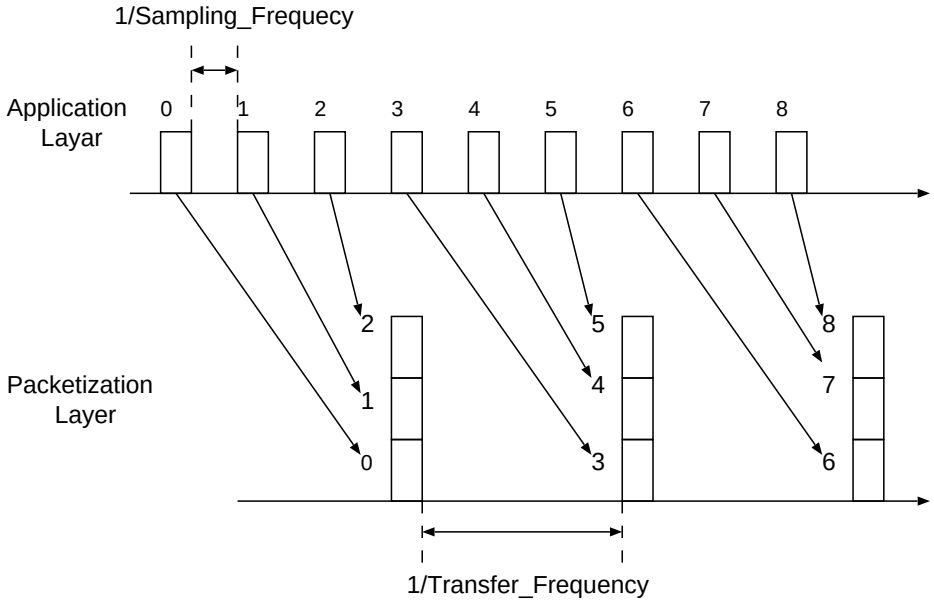


図 4.3: パケット化における一般モデル

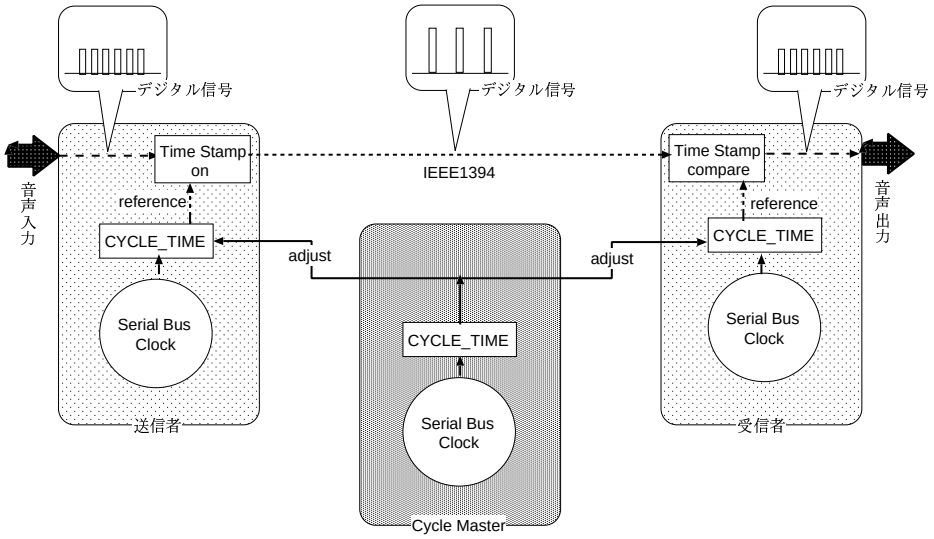


図 4.4: mLAN におけるクロック伝送による同期機構

別のシステムを用いる必要がある．そのため，mLAN と映像転送システムの同期処理が何らかの方法で必要となる．

4.4 長野オリンピック開会式

1998 年の長野オリンピック開会式では，5 大陸の都市 (ニューヨーク，北京，シドニー，ベルリン，ケープタウン) を衛星回線で結んでの『第九』合唱が行われた (以下長野オリンピック

方式).

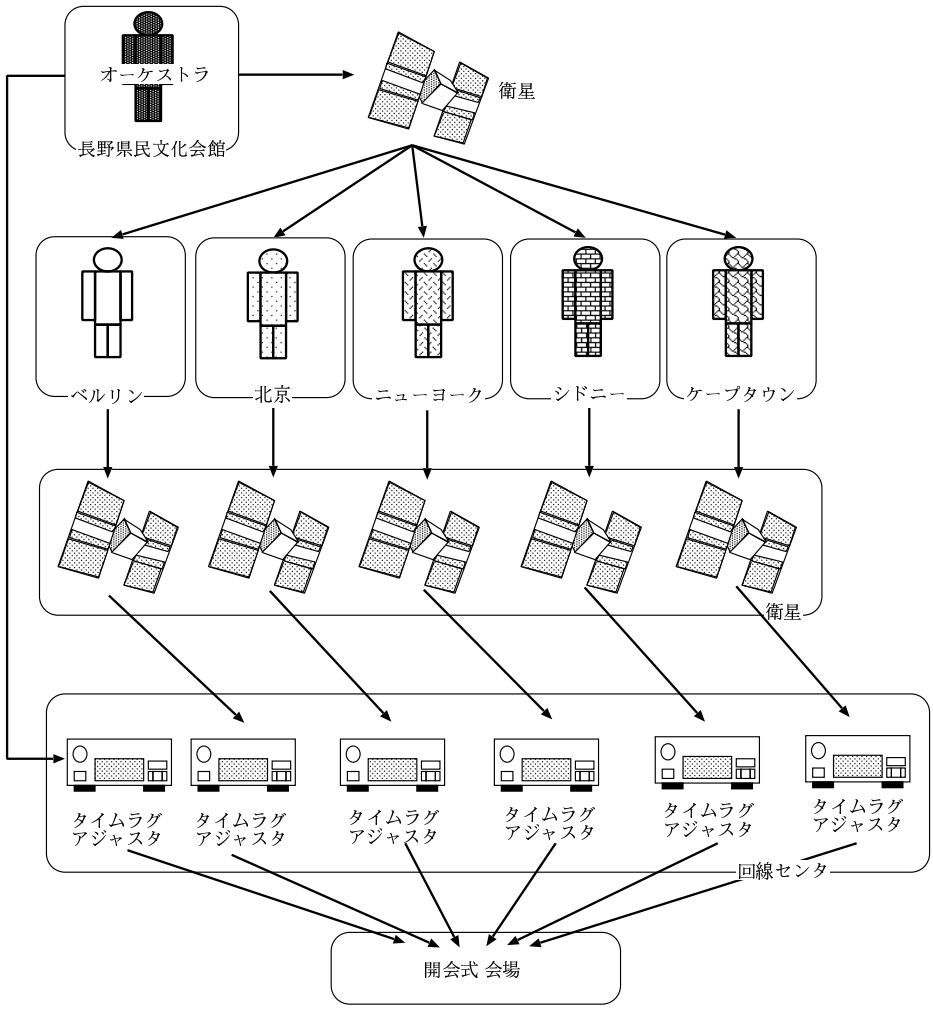


図 4.5: 長野オリンピック 開会式概要

4.4.1 長野オリンピック方式の特徴

長野オリンピック開会式のシステム概要を図 4.5に示す．指揮者，オーケストラが演奏する長野県民文化会館が拠点となる．衛星回線を経由して各国の合唱隊，及び開会式会場の隣に設置された回線センタへと ISDN 回線経由で指揮者とオーケストラの映像・音声が転送される．各国の合唱隊は文化会館から送られてきた映像・音声を基に合唱を行う．収録された映像・音声は再度衛星回線を経由し，回線センタへと収容される．

それぞれのストリームは伝送距離や伝送経路の違いにより，到達時間のずれが発生する．開会式では，到達時間のずれをもっとも遅延時間の長い拠点に合わせて出力するためにタイムラグアジャスタ [23] が用いられた．タイムラグアジャスタは，メモリに映像 (NTSC)・音声信号を蓄積，設定値に応じて遅延させた信号を出力する．タイムラグアジャスタを用いることにより，最大 5 秒間の映像データ (NTSC 29.97fps) 蓄積，及び 1msec 単位での遅延時間設定が可能

である．タイムラグアジャスタを用い，遅延時間の長かった北京の 1938msec に微調整に必要な遅延時間を考慮した 133msec を加算した 2071msec を基準値とし，他の音声ストリームに対しての遅延時間設定を行った．

4.4.2　長野オリンピック方式の検証

長野オリンピック方式は複数のストリームを扱う点，それぞれのストリームは同期が取られた上で視聴者に届く点，合唱の基準となる指揮者及びオーケストラの映像・音声が合唱者間で共有される点の 3 点において本システムと類似している．しかしベルリンの合唱隊が北京の音声を聞くといった，拠点間の音声データの聴取は考慮されていないため，演奏者が和音やダイナミックスを確認することは困難である．

4.5　まとめ：インターネットを用いた音楽の共同制作

本章ではネットワークを用いた音楽の共同制作のシステム事例として坂本龍一 MIDI ライブ，長野オリンピック開会式の紹介と優位点，問題点について述べた．その結果，第 2章で述べた音楽共同制作環境構築の際の要件のうち，遅延時間の管理はクライアントによるバッファリング (坂本龍一 MIDI 方式)，タイムラグアジャスタの仕様 (長野オリンピック方式) などの手法を用いて行われている．しかし他の演奏者の音声聴取，指揮者映像などの送信によるタイミング共有の 2 つを同時に満たしてはいない．

次章ではこれまで述べてきた事柄を元に，本モデルの設計について述べる．

第5章　設計：オーケストラ演奏機構

第 4章において，既存のネットワークを用いた音楽共同制作の問題を述べた．以下に 3 つの課題を整理する．

1. 他の演奏者の音声の聴取 (50ms - 80ms 以内のフィードバック)
2. 拠点毎に異なる遅延時間の取得
3. 演奏者間におけるタイミングの共有

5.1　設計概要

本研究の設計概要を図 5.1に示す．

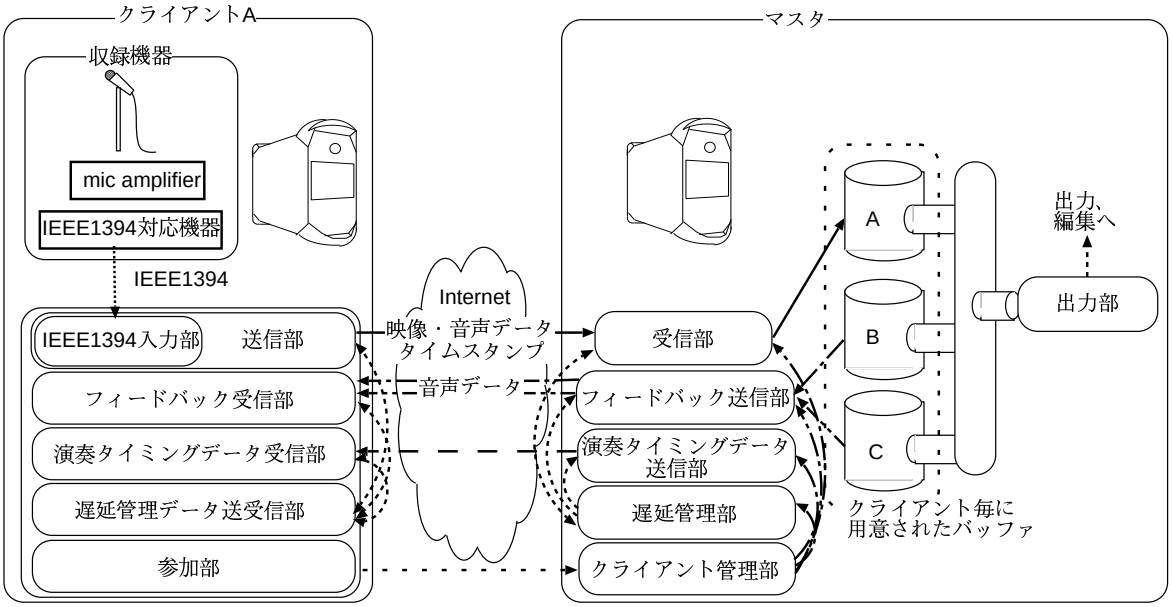


図 5.1: 設計概要

本論文では各演奏者の映像・音声データを受信，遅延時間などの管理などを行う転送機器をマスタ，演奏者の用い，マスタに映像・音声データを転送する転送機器をクライアントと呼ぶ．
以下にそれぞれの詳細を示す．

1. クライアントデータベース
   演奏を行うクライアント情報の管理を行う．

2. 遅延管理データ送受信部・遅延管理部
   映像・音声データに含まれるタイムスタンプの管理，解析，バッファ管理を行う．

3. クライアント：映像・音声データ送信部
   マスタへの音声，または映像・音声の転送を行う．

4. マスタ：映像・音声データ受信部
   クライアントからの音声，または映像・音声の受信を行う．

5. フィードバック送信部・受信部
   演奏者に他の演奏者の音声転送を行う．

6. タイミングデータ送信部・受信部
   演奏のタイミング，リズムを合わせるための映像・音声転送を行う．具体的には指揮者映像の映像などが想定される．

合奏の流れは以下に示される 7 つの手順によって行われる．

1. クライアント：マスタへ合奏参加要求の送信

2. マスタ：クライアントへ音声データ送信用ポート番号を告知

3. マスタ：クライアントへタイミング信号を送信

4. クライアント：タイミング信号を基に演奏者が演奏を開始

5. クライアント：マスタへの音声，または映像・音声データ転送

6. マスタ：各クライアントへ他のクライアントから受信した音声データの送信

7. クライアント：フィードバックの受信，再生

クライアントにおいてタイミングデータの再生と，フィードバックの再生までの遅延時間が 80ms 以内であれば合奏可能となる．この要求時間を満たすにはタイミングデータ送信のタイミングが重要となる．複数拠点を結んだ合奏を行う場合，演奏開始時間を同期を取る手法がある．演奏開始時間の同期を取った場合と取らない場合の概念図を図 5.2に示す．

演奏開始時間の同期を取った場合，ネットワーク伝送遅延の揺らぎを考慮する必要はあるが，バッファサイズが同期を取らない場合に比べ小さく済むため，リアルタイム性を確保しやすいというメリットがある．しかしフィードバックを可能な限り早く転送する必要のある本システムでは，ネットワーク伝送遅延の大きなクライアントに併せて，遅延の小さなクライアントまでもが影響を受け，フィードバック転送にかかる時間が増加する．第 2.2節の演奏可能遅延許容時間の式より，最もネットワーク伝送遅延の大きなクライアント C(ネットワーク伝送遅延は往復で同一と仮定) が，他のクライアントのフィードバックを受信するのにかかる時間は，演奏開始時間の同期を行う場合ではマスタが受信する時間は全てのクライアントにおいて同一だと仮定すると，[クライアント A, B のマスタへの転送時間 40ms] + [マスタからクライアントへの転送時間 40ms] より 80ms かかる (図 5.2上)．同期を行わない場合は [クライアント B からマスタへの転送時間 20ms] + [マスタからクライアント C への転送時間 40ms] より 60ms かかる (図 5.2下)．このことより，バッファリングのコストは高くなるが，遅延許容時間の小さなフィードバックの転送に有利な後者を本研究では用いる．

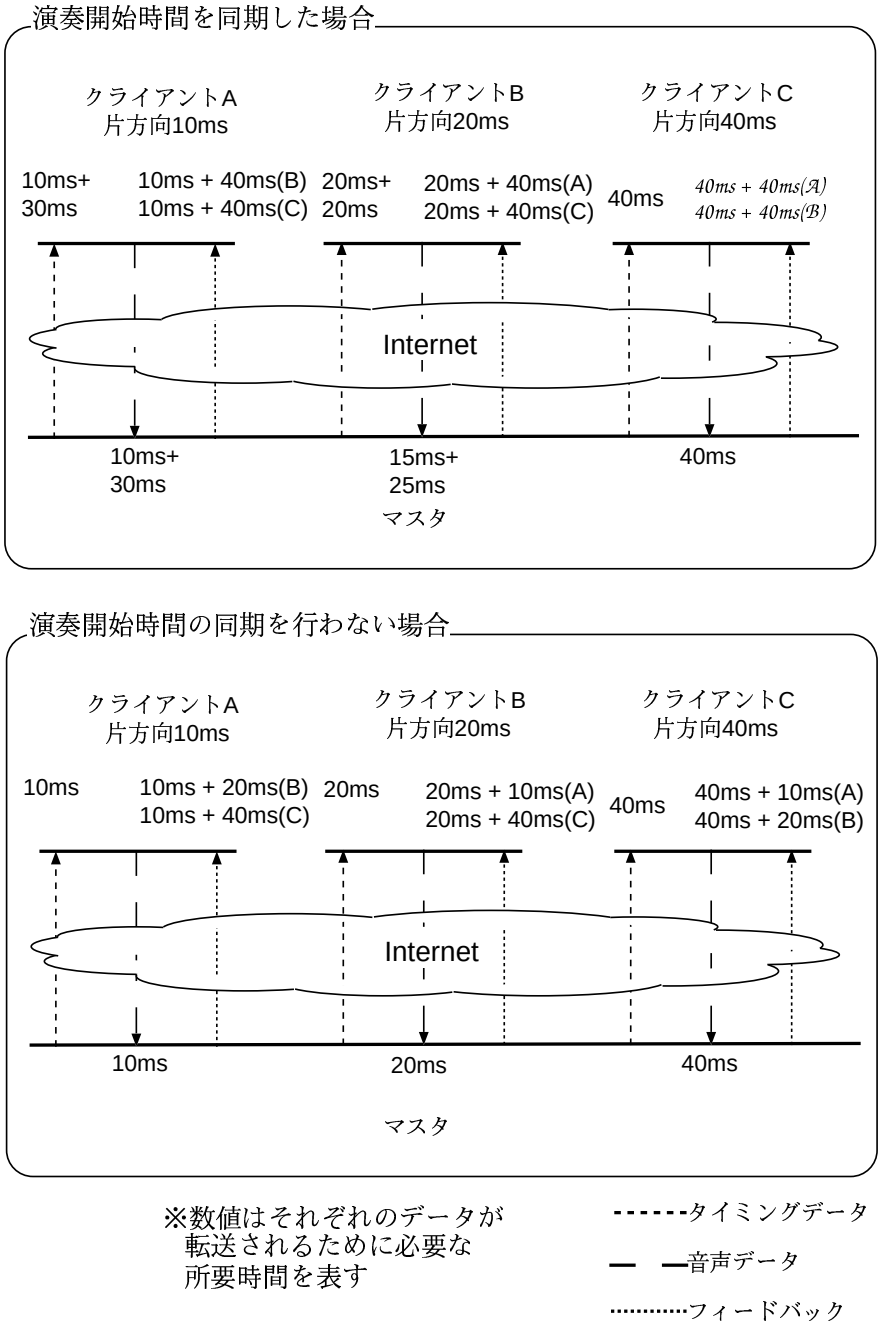


図 5.2: 演奏開始時間の同期と各データの転送所要時間概念図

5.2 映像・音声入出力

本機構ではマスタにおけるタイミングデータ読み込み，クライアントにおける映像・音声データ読み込み，タイミングデータ，フィードバックデータの書き出しを外部デバイスに対して行い，演奏者や編集機器に伝送する必要がある．データ欠損に伴う合奏の品質低下を防ぐために，安定したデータ入出力がなされる必要がある．

マルチメディア転送において多用される外部デバイスの一つに，IEEE1394[11] が存在する．

IEEE1394 からの映像・音声データの入出力に Isochronous 転送を用いることにより，125 μ秒毎にデータ転送の優先権が与えられ，リアルタイム性の高いデータ入出力を行うことができる．Isochronous packet の詳細は後述する．

マスタにおけるタイミングデータ入力，クライアントにおける映像・音声データ入力，及びマスタより受信したタイミングデータ，フィードバックデータ出力は第??節でも述べた IEEE1394 デバイスを介した Isochronous 転送によって行う．画像転送を踏まえ，本研究では DV を用いる．

5.3 クライアント管理データベース

マスタは，演奏に参加するクライアント情報を管理する必要がある．クライアントからの接続要求を受けたマスタは，クライアントに映像・音声データ送信のためのポート番号を通知する．同時に以下の 3 つのクライアント情報をデータベースとして作成，保持する．

- IP アドレス
- 映像・音声データ用 ポート番号
- プライオリティ

5.4 遅延時間の管理

複数拠点からマスタへ送信される音声データは，同期機構により同時到達性が確保された状態でミキサに送出される必要がある．そのため同期機構で扱われるタイムスタンプの信頼性が重要となる．

タイムスタンプとして，NTP を参照，NTP の指し示す値が複数拠点において同一であると仮定した場合，複数地点における同期が実現される．NTP は原子時計，GPS，無線時計を参照するプライマリサーバ (stratum 1) から最大 15 サーバ (stratum15) までの階層構造による参照が可能である [24] が，下の階層に位置するサーバを参照するほど時刻の同期精度は低くなる．クライアントが必ずしも信頼性の高い NTP サーバを参照にできる環境にあるとは限らない．

本機構では，映像・音声データ同期には独自の相対時間を用い，ネットワーク遅延時間を計測するためにマスタで取得された時間を基準とする絶対時間を用いる．これによりマスタを基準とする時間軸の管理を実現する．以下に映像・音声データ同期のための相対時間によるタイムスタンプと，ネットワーク遅延時間の計測のための絶対時間によるタイムスタンプについて述べる．

映像・音声データ同期タイムスタンプ

音声，もしくは映像・音声データ同期のために相対時間から求められるタイムスタンプを用いる．後述するフィードバック到達時に送信データをリセット，カウントを開始する．これにより，映像・音声データ同期タイムスタンプカウントは演奏を行う全てのクライアントにおいて，同時に開始される．

ネットワーク遅延時間計測タイムスタンプ

後述するバッファサイズ，タイミングデータ，フィードバックの送信のために，クライアント・マスタ間の遅延時間を知る必要がある．

マスタ-クライアント間の遅延時間 (MC) は，タイミングデータのクライアント受信時間 (ts_1) とマスタの音声データ受信時間 (ts_2) より

$$MC = ts_1 - ts_2$$

によって求められる． タイミングデータ内における拡張 RTP に送信時にマスタで取得された時間を挿入．クライアントはマスタの時間をそのまま送り返すことで，マスタ側音声データ受信時のマスタ CPU 時間 (マスタ時間) との比較を行うことにより伝送遅延時間を取得する．拡張 RTP の詳細は後述する．

これらの値を参考値とし，後述するバッファのサイズを決定する．クライアントから送信される映像・音声データ用バッファは，演奏開始前に用意される必要があることから，演奏に用いられる相対時間ではなく，マスタの時間を軸としたネットワーク伝送遅延の測定を行う．

遅延時間計測の結果，パケットロスの発生，ネットワーク遅延時間の揺らぎによるパケット到着時間の揺らぎを確認できる．演奏開始前に決定したバッファサイズで到着時間の揺らぎを吸収できない場合にはリアルタイム性を重視し，該当する相対時間のパケットのバッファへの書き込みを見送る．

収録環境による遅延時間計測

クライアントにおいて，タイミングデータ受信時のタイムスタンプ (ts_1) と，音声データ転送時のタイムスタンプ (ts_3) の差分をとることで，転送機器内の処理を含めた収録環境による遅延時間 (CIP) は

$$CIP = ts_3 - ts_1$$

によって求められる．

OS のタスクスケジューリングなどの要因によって遅延時間の遅れが大幅に確認された場合，マスタ側に警告信号が出される．警告信号を受信したマスタは該当するクライアントの音声データをフィードバックとして送信することの一時中止などの対応を行う．

5.4.1 RTP フォーマットの拡張

音声データ，遅延時間計測の際に必要な項目を以下にあげる．

1. データ到達順序確認用シーケンス番号
2. ネットワーク遅延計測用タイムスタンプ
3. 音声データ同期用タイムスタンプ
4. マスタ-クライアント間のネットワ-ク遅延時間
5. 収録環境による遅延時間

これらのうち，1，2 は第 3.1.2節で述べた RTP に備わった機能である．実装のしやすさを踏まえ，本研究では RTP フォーマットの拡張による実装を行う．本研究で実装する拡張 RTP フォーマットを図 5.3に示す．

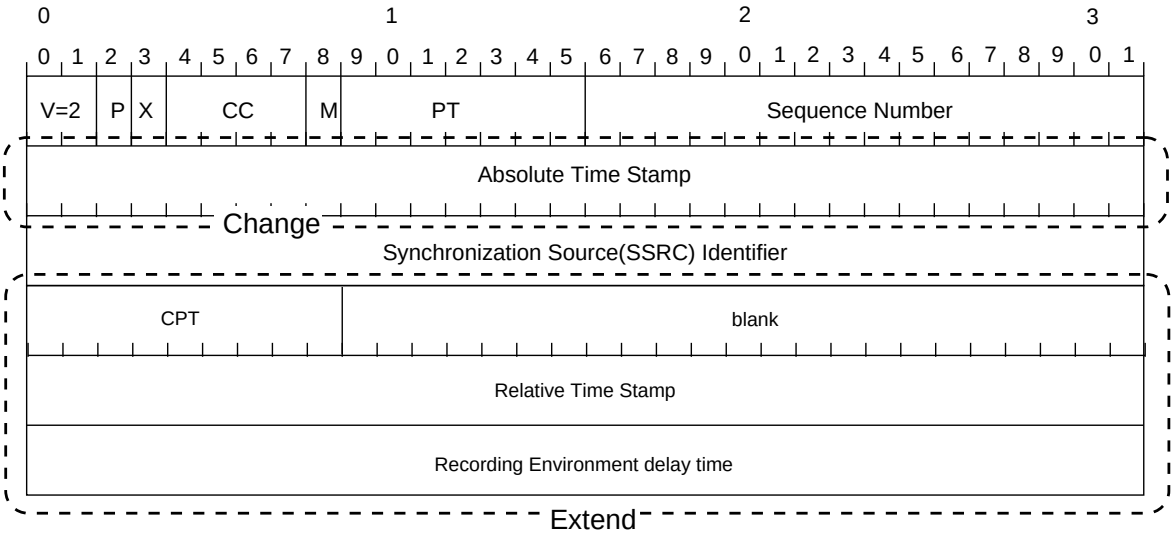


図 5.3: 本研究における拡張 RTP フォーマット

相対時間によるタイムスタンプを図 5.3中，Relative Time Stamp(32bit) に格納する．クライアント-サーバ間のネットワーク遅延時間は図中 CMt に格納される．5.4項で述べたクライアント収録環境による遅延時間 (32bit) を図中 Recording Environment delay time に格納する．図中 CPT にはクライアント，マスタ双方の挙動を指示するためのフラグが立つ．

5.4.2 マスタ：バッファリング

バッファは以下の遅延吸収機能を持つ必要がある．

1. ネットワーク伝送遅延時間吸収
   ネットワーク上の異なる地点に存在する複数のクライアントへの対応
2. 収録環境による遅延時間吸収
   クライアント毎に異なる収録環境への対応
3. 1, 2 の遅延時間の揺らぎ吸収

バッファサイズの決定のためには上記の遅延時間の取得が必要となる．なお，同期処理については遅延時間の管理をすることで既存の同期技術の利用ができるため，本論文では対象外とする．

バッファ管理

前述したバッファの遅延吸収機能の要求より，バッファはクライアント毎に用意され，クライアントから送信された音声データが格納される．

バッファ内の映像・音声データは本研究の拡張 RTP フォーマット 5.3内の相対時間を基に扱われ，ソフトウェアミキサと連携がなされた上で出力される．

5.5 タイミングデータ送信部・受信部

演奏タイミングデータとしては，指揮者などの映像を用いることを想定する。タイミングデータには第 5.4.1項で述べた拡張 RTP フォーマットを用い，送信時に取得された時間が記録される．

演奏開始時に拡張 RTP フォーマット (図 5.3) の CPT に演奏開始フラグが立つ．

受信した演奏タイミングデータの拡張 RTP フォーマット内 (図 5.3)，CPT において演奏開始フラグが確認された場合，相対時間をデータベースに記録する．受信した映像・音声データは IEEE1394 に書き出される．

5.6 映像・音声データ送受信部

5.6.1 クライアント：映像・音声データ送信部

送信部は IEEE1394 からデータを受信する部分と，IP ネットワークに対してデータを送信する部分に分かれる．送信部では音声データ，または必要に応じて画像データの送信を送信できるものとする．

クライアントは IEEE1394 より Isochronous 転送を用いて，IEEE1394 Isochronous packet を受信する．

IP ネットワークへのデータ送信を行う部分では，送信用バッファの先頭に第 5.4.1項で述べた拡張 RTP フォーマットを付加し，送信用バッファに可能な限りデータを挿入する．

拡張 RTP フォーマットのシーケンス番号 (Sequence Number) は IP パケット毎に 1 ずつ増加する．絶対時間タイムスタンプ (Absolute Time Stamp) はタイミングデータに存在するマスタ時間が挿入される．一方，相対時間タイムスタンプ (Relative Time Stamp) はマスタから送信されるタイミング信号に起因する演奏開始時より，1 ずつ増加する．

Recording Environment delay time にはクライアント側においてタイミングデータの受信時にクライアントで取得された時間と，音声データの送信時間にクライアントで取得された時間の差分が記録される．この差分の結果，大幅な遅延時間が観測された場合には，クライアント側転送機器，収録環境において OS のタイムスケジューリングなどに異変が起きたと見なされ，OVERTIME などの識別子が挿入される．識別子の挿入により，マスタは識別子があるときのみに適切な処理を行う，という処理の単純化ができる．識別子によってダイアログの表示によるユーザへの通知，該当するクライアントのバッファリング一時停止などの処理を行う．

クライアントは音声データの送信を映像・音声データと音声データのみの 2 つのストリーム送信を行う．マスタは映像・音声データと音声データの 2 つを個別に管理することにより，後述する音声フィードバックの効率化を行う．

5.6.2 マスタ：映像・音声データ受信部

受信部は IP ネットワークからデータを受信する部分と，前述したバッファにデータを書き込みする部分に分かれる．

受信したデータをクライアント毎に用意したバッファ，及びクライアントデータベースにコピーする．それぞれのコピーを図 5.4に示す．各クライアントより受信したデータに含まれる IEEE1394 Isochronous パケットは各クライアント毎に用意されたバッファにコピーされる．また，拡張 RTP フォーマット内に含まれるマスタ→クライアント間のネットワーク伝送遅延時間，及び受信時のタイムスタンプより算出されるクライアント-¿マスタ間のネットワーク伝送遅延時間は各クライアント毎に用意されたクライアントデータベースにコピーされる．

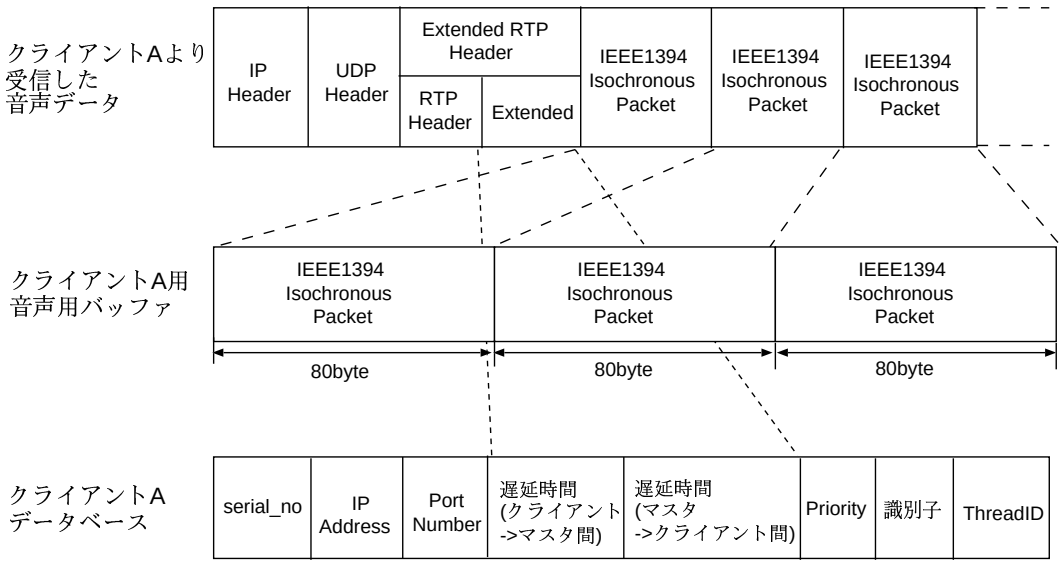


図 5.4: IP パケット受信からバッファ，データベースへのコピー

5.7 フィードバック送信部・受信部

フィードバックの転送は遅延許容範囲時間内に納める必要があるため，同期処理を行わず，各クライアントに即座に送信される．また，クライアントにおける音声データ送信とフィードバック受信時の遅延時間，及びクライアントが複数のフィードバックデータを受信することを踏まえ，エンコード/デコードの時間が短い音声データのみを送信の対象とする．

マスタにおいて，フィードバック送信用スレッドは各クライアント毎に作成される．各クライアント毎のバッファから音声データのみを読み出し，クライアントに送信する．

フィードバック受信用スレッドは音声データ送信スレッドとは別のスレッドとして作成される．受信した音声データは IEEE1394 へ書き出される．

5.8 実装における課題の整理

以上に述べたことから，実装の際に配慮する必要がある項目を以下に述べる．

- 安定した映像・音声データ入出力
  IEEE1394 を利用した映像・音声データの入出力，及びネットワーク転送を行う．

- 内部処理
  タイミングデータ送受信，映像・音声データ送受信，フィードバックデータ送受信といった各部位はアクセスするリソースが IEEE1394 デバイス，バッファといったように異なるため自律して動作する必要がある．また，マスタはクライアント数分の処理を行う必要があるため，柔軟な各部位の増減ができる内部処理方法が必要である．

- クライアント管理データベース
  クライアント毎に異なる IP アドレス，バッファなどの情報に対し，各モジュールからアクセスすることのできる実装が必要である．

- 遅延時間管理
  タイミングデータ送受信，映像・音声データ送受信，フィードバックデータ送受信時に絶対時間によるタイムスタンプを取得する必要がある．タイムスタンプの粒度はミリ秒といった細かな単位で行う．

5.9 まとめ：オーケストラ演奏機構の設計

本章では第 2章において提案されたモデルに基づいた設計について述べた．クライアントデータベース，遅延管理データ送受信部，映像・音声データ送受信部，フィードバック送受信部，タイミングデータ送受信部の 5 つのモジュールについて設計を行った．次章では本章で述べた設計を基に行った実装について述べる．

第6章　実装：オーケストラ演奏機構の構築

6.1　実装概要

本章では第 5章の設計に基づいた実装に関して述べる．
表 6.1に実装に用いたソフトウェア環境を示す．

表 6.1: 実装ソフトウェア環境

OS	MacOS 10.3.2
プログラミング環境	Xcode 1.1.0 Interface Builder 2.4 QuickTime 6.5
API	Carbon
コンパイラ	gcc 3.3
プログラミング言語	C 言語

6.2　DVTS

IEEE1394 からデータを取得，IP データグラム化を行い，ネットワーク越しに映像・音声データ転送を行うものに DVTS が存在する．DVTS は以下の特徴を持つ．

本研究では DVTS(Digital Video Transport System)[25][26] を母体とした実装を行う．以下に DVTS の概要，実装に用いるメリットを述べる．

- IEEE1394 からの映像・音声データの入出力に Isochronous 転送を用いる
- 音声は IEEE1394 からの入力を非圧縮で転送する
- IPv4/IPv6 に対応する
- RTP を利用した通信を行う
- Unix 用ソフトウェアとして，Open Source で提供されている

これらの特徴より，本研究はリアルタイム性が高く，ソースコードが公開されているため開発の行いやすい DVTS の機能拡張という形で実装を行う．

6.2.1 DVTS for MacOSX

本研究では DVTS for MacOSX[27] を用いた実装を行う．従来の DVTS と違い，DVTS for MacOSX では IEEE1394 からの映像・音声データの読み込み，書き込み，転送などの処理が独立したスレッドで処理されているため，スレッドを追加することによる機能拡張が可能である．従来の DVTS と DVTS for MacOSX の内部処理比較を図 6.1に示す．

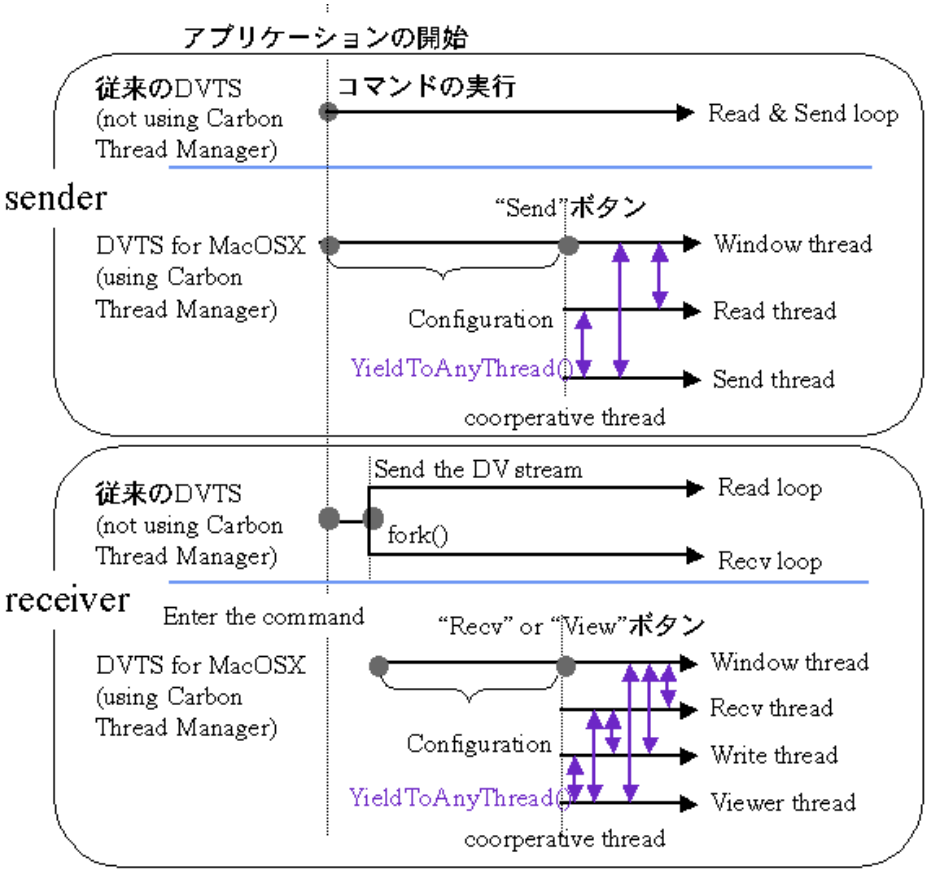


図 6.1: 従来の DVTS と DVTS for MacOSX

6.3 内部処理：スレッド

各部位は独立したスレッドとして作成される．Carbon Toolbox の 1 つである Thread Manager[28] を用いた協調スレッドとして処理される．Thread Manager は pthread をベースに Apple 社が開発・機能向上させたものであり，Quick Time API を始めとする Carbon API をスレッドとして扱うことができる．同時にメモリ管理，CPU リソースの管理も適切な状態で行われる．

マスタにおける Thread Manager 適用例を図 6.2に示す．各スレッドはループ内で定期的に Thread Manager を呼ぶ．Thread Manager はリソースを管理し，適切なスレッドに制御権を譲渡する．アプリケーション開始時には GUI を管理するスレッドが発生する．クライアント接続を管理するスレッド発生後，登録されたクライアントの数だけタイミングデータ送信部，音声データ受信部，フィードバック送信部が発生する．クライアントにおけるスレッドも同様に発

生する．またMacOSXではBSD socketを拡張，GUI，スレッドとの親和性を高めたCFSocket を用いるため，CFSocketの管理を行うSocket Managerが発生する．

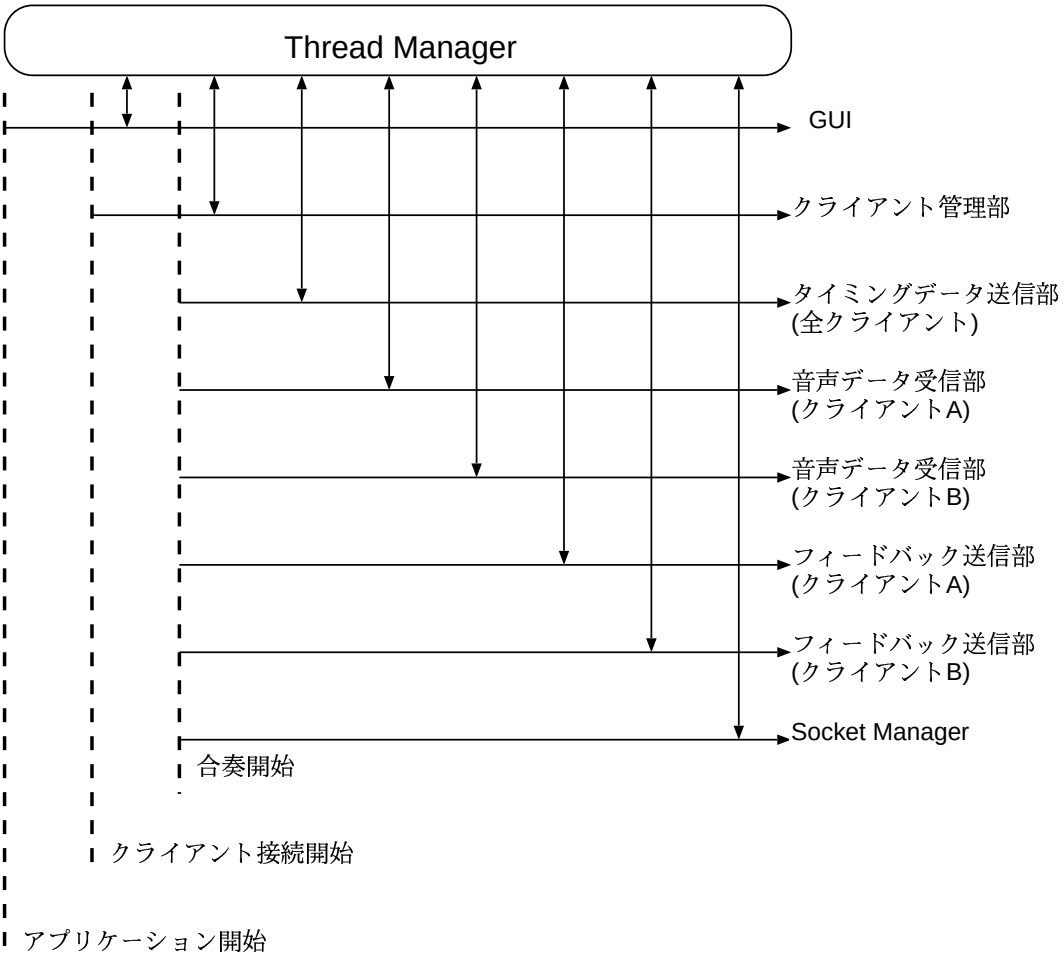


図 6.2: Thread Managerによるマスタ スレッド群の管理

本機構はタイミングデータ送信スレッドとクライアント数分の映像・音声データ受信スレッド，フィードバックデータ送信スレッドの作成を行う．スレッドは後述するdb_entry構造体配列を介することによってクライアント毎の情報管理を行う．db_entry構造体配列にアクセスし，任意のクライアント情報を呼び出す引数として，シリアル番号が渡される．任意のクライアントに対するアクセスを図6.3に示す．

6.4 参加部・クライアント管理部

本実装参加部・クライアント管理部の機能を以下に示す．

- TCP socketの作成
  合奏参加要求パケットが偶発的に落ちることを防止するため，再送機能を持つTCPを用いる．

```
int i;

for(i=0; i > dvrecv_param->client_seq; i++){
   (省略)
   /****************** add Threads **********************/
   /*各クライアントからの音声データ受信用スレッド*/
   err = AddConcertinoRTPReadThread(db_entry, i);
   if(err != noErr){
      return err;
   }
   /*各クライアントへのフィードバックデータ送信スレッド*/
   err = AddConcertinoFeedbackThread(db_entry, i);
   if(err != noErr){
      return err;
   }
   (省略)
}
```

図 6.3: 各スレッドの作成と引数

- クライアント:マスタへの接続要求データ送信
- マスタ:クライアントへ音声データ転送用ポート番号の告知
- マスタ:クライアント情報の登録，保持
  各クライアントとやり取りを行うための情報を後述する db_entry 構造体配列に登録を行う．

6.4.1 マスタ：クライアント情報の管理

本実装では複数のクライアントに対し，管理，データの送受信を行う必要がある．クライアントの一括管理のために db_entry 構造体を独自に定義する．db_entry 構造体を図 6.4に示す．db_entry 構造体は構造体配列を用いることにより，クライアントの IP アドレス，演奏データ送信時のポート番号，共有メモリ ID，各種スレッド ID，タイムスタンプなどの管理を行う．

クライアント情報へのアクセス例を図 6.5に示す．db_entry 構造体配列にクライアントのシリアル番号を与えることによりアクセスする．

6.4.2 マスタ：クライアントからの接続要求処理

接続要求受信の際に取得・設定が必要な情報は IP アドレス，音声データ送受信用ポート番号，シリアル番号がある．シリアル番号とはクライアントを識別するために用いられる通し番号であり，第 6.4項で述べた db_entry 構造体配列にアクセスするために用いられる．クライアント

```
struct db_entry{
    int serial_no; /* シリアル番号 */
    int port;   /* 映像・音声受信用ポート番号 */
    int priority; /* プライオリティ設定 */
    char nick;  /* 識別子 */
    /********* 共有メモリ *********/
    int video_shmid; /* 映像用共有メモリ ID */
    int audio_shmid; /* 音声用共有メモリ ID */
    /********* フレームバッファ *********/
    int frames_in_buffer;
    u_int32_t *last_dv_buffer;
    u_int32_t *video_buf;
    struct shm_frame *video_framebuf;
    u_int32_t *audio_buf;
    struct audio_shm_frame *audio_framebuf;
    struct static_video_frame *static_video_frame;
    /********* ソケット *********/
    int timing_sock /* タイミングデータ用ソケットディスクリプタ */
    int feedback_sock /* フィードバックデータ用
                                    ソケットディスクリプタ */
    char *conreq; /* クライアント IP アドレス */
    struct sockaddr_in timing_sin; /* タイミングデータ用 */
    struct sockaddr_in feedback_sin; /* フィードバックデータ用 */
    /********* スレッド ID *********/
    ThreadID    RTPReadThreadTID; /* 映像・音声データ受信用スレッド ID */
    ThreadID    TimingThreadTID; /* タイミングデータ送信用スレッド ID */
    ThreadID    FeedbackThreadTID; /* フィードバック送信用スレッド ID */
    /********* time stamp *********/
    u_int32_t timing_send_ts;/* タイミングデータ送信時タイムスタンプ */
    u_int32_t data_recv_ts; /* 音声データ受信時タイムスタンプ */
    u_int32_t network_ts; /* ネットワーク伝送遅延 */
    u_int32_t transmission_ts; /* 伝送遅延時間
                                  (収録環境による遅延時間含)*/
    u_int32_t feedback_send_ts;/* フィードバック送信時タイムスタンプ */
    u_int32_t client_ts; /* クライアント側転送処理時間 */
};
```

図 6.4: db_entry 構造体配列

の接続要求から IP アドレスなどの情報を db_entry 構造体配列に登録する処理の流れを 6.6に示す.

```
int n;
int i;
(省略)
/* 全エントリのクライアントに対してのデータ送信 */
for (i=0; i <= dvrecv_param->client_seq; i++){
  n = sendto(db_entry[i].timing_sock,
             buf, buflen, 0,
             (struct sockaddr *)&db_entry[i].timing_sin,
             sizeof(db_entry[i].timing_sin));
  if (n < 1) {
      perror("timing_send_pkt : sendto");
  }
  db_entry[i].timing_pkt_count++;
}
(省略)
```

図 6.5: クライアント情報の取得

6.5 遅延時間の管理

記録された時間は後述する拡張 RTP フォーマットに unsigned int 型で格納される．整数値化された時間を挿入するために gettimeofday() から得られる秒，マイクロ秒の値を加算，乗算して整数にする関数 ConcertinoGetCurrentTime() を作成した．

6.5.1 マスタ：往復に要する遅延時間の管理

マスタはタイミングデータ送信時，映像・音声データ受信時に ConcertinoGetCurrentTime() の呼出しを行う．タイミングデータ送信時に取得された時間データは db_entry 構造体配列内の timing_send_ts 変数内に格納される．映像・音声データ受信時にシリアル番号の確認を行った後に ConcertinoGetCurrentTime() を呼出し，db_entry 構造体配列内の data_recv_ts 変数に格納する．data_recv_ts と timing_send_ts 変数の比較することで伝送遅延時間の取得，db_entry 構造体配列内の transmission_ts 変数への登録を行う．

transmission_ts と後述するクライアント収録環境の遅延時間との差を取ることで，ネットワーク伝送遅延時間 (db_entry 内 network_ts 変数) を取得することができる．network_ts 変数を定期的に知ることにより，ネットワーク伝送遅延時間の揺らぎを感知できる．

6.5.2 クライアント：収録環境による遅延時間の管理

クライアントはタイミングデータ受信時，音声データ送信時に ConcertinoGetCurrentTime() の呼出しを行う．タイミングデータ受信時に取得された時間データは dvsend_param 構造体内の timing_recv_ts 変数に格納される．映像・音声データ送信時には ConcertinoGetCurrentTime() を呼出し，timing_recv_ts との差を求めることで内部処理に要した時間が求められる．求めら

```
if( dvrecv_param.client_seq == 0 ){
    printf("Connected from Client %d\n", dvrecv_param.client_seq );
    (省略)
    /* クライアントシリアル番号の登録 */
    db_entry[dvrecv_param.client_seq].serial_no =
                                        dvrecv_param.client_seq;

    /* 映像・音声データ転送用ポート番号の登録 */
    db_entry[dvrecv_param.client_seq].port = port_no;

    /* クライアント IP アドレスの登録 */
    db_entry[dvrecv_param.client_seq].conreq =
inet_ntoa(client.sin_addr);

    /* クライアントへの映像・音声データ転送用ポート番号の通知 */
    snprintf(buf, sizeof(buf), "%d", port_no);
    write(accepted, buf, strlen(buf));
    close(accepted);

    /* クライアントシリアル番号を加算 */
    dvrecv_param.client_seq++;

    /* 次に接続するクライアントの映像・音声データ転送用ポート番号 */
    port_no=port_no + 4;
}
```

図 6.6: クライアント接続要求からの情報取得と登録処理

れた値は拡張 RTP フォーマット内 client_ts 変数に格納され，次回の映像・音声データ送信時に送信される．

6.6 拡張 RTP フォーマット

本実装で用いる拡張 RTP フォーマットを図 6.7に示す．拡張 RTP フォーマットはタイミングデータ，映像・音声データ，フィードバックデータの 3 つのデータに利用される．

6.7 バッファリング

マスタはクライアント毎に映像・音声のバッファリングを行う．バッファは映像・音声データと，音声データのみの 2 つを用意する．これにより後述するフィードバックの際に映像・音声データから音声データを取り出す処理が省かれるため，効率的な処理が可能となる．それぞ

```
typedef struct {
   unsigned int version:2;   /* プロトコルバージョン */
   unsigned int p:1;         /* パディングフラグ */
   unsigned int x:1;         /* ヘッダ拡張フラグ */
   unsigned int cc:4;        /* CSRC カウント */
   unsigned int m:1;         /* マーカビット */
   unsigned int pt:7;        /* ペイロードタイプ */
   unsigned int seq:16;      /* シーケンス番号 */
   u_int32 ts;               /* マスタ時間タイムスタンプ */
   u_int32 ssrc;             /* シンクロナイゼーションソース */
/* u_int32 csrc[1];     */   /* CSRC(オプション) */
   /***** RTP 拡張部分 *****/
   unsigned int con_pt:8;    /* 拡張ペイロードタイプ */
   unsigned int option:24;   /* オプション */
   u_int32 relative_ts;      /* クライアント：相対時間タイムスタンプ */
   u_int32 client_ts;        /* クライアント：処理時間 */
} rtp_hdr_t;
```

図 6.7: 拡張 RTP ヘッダ

れのバッファはフィードバックデータ送信，ミキサへの出力を考え，共有メモリの形で用意される．

共有メモリ作成のための関数 concertino_attach_shared_memory() は，shmat() 関数を用いて共有メモリを作成後，db_entry 構造体配列に共有メモリ ID の登録を行う．db_entry 構造体配列を用いた共有メモリの用意を図 6.8に示す．共有メモリに対する書き込み，読み込みの際は db_entry 構造体配列を経由して行う．

6.8 タイミングデータ送信部・受信部

6.8.1 マスタ：IEEE1394 からのデータ受信

IEEE1394 をオープンし，タイミングデータに用いられる映像・音声データの読み込みを行う．

本実装では DVTS for MacOSX で用意されている prepare_ieee1394() を用いた IEEE1394 デバイスのオープン，main_loop() を用いた IEEE1394 デバイスからの読み込みを行う．

6.8.2 マスタ：IP ネットワークへのデータ送信

タイミングデータは，全クライアントに対して同一のデータが送信されるため，クライアント毎にスレッドを行う必要がない．前述した db_entry 構造体配列を利用した映像・音声データの一括転送を行う．演奏タイミングデータの送信を図 6.9に示す．シリアル番号の大小による送信時間の差異は伝送遅延として他部により吸収される．

```
(省略)
int concertino_prepare_shared_memory (int i)
{
  int video_shm_size, audio_shm_size;
  int video_shmid, audio_shmid;

  /* バッファサイズの取得 */
  video_shm_size = db_entry[i]->frames_in_buffer *
                (sizeof(struct shm_framebuf) +
                sizeof(struct static_video_frame));
  audio_shm_size = db_entry[i]->frames_in_buffer *
                (sizeof(struct audio_shm_framebuf));

  /* 映像用共有メモリ ID の取得 */
  if ((video_shmid =
    shmget(IPC_PRIVATE, video_shm_size, 0600)) < 0) {
    perror("shmget in prepare_shared_memory");
    return(-1);
  }
  /* 音声用共有メモリ ID の取得 */
  if ((audio_shmid =
    shmget(IPC_PRIVATE, audio_shm_size, 0600)) < 0) {
    perror("shmget audio in prepare_shared_memory");
    return(-1);
  }

  /* 各共有メモリ ID を db_entry 構造体へ登録 */
  db_entry[i].video_shmid = video_shmid;
  db_entry[i].audio_shmid = audio_shmid;

  return(1);
}
```

図 6.8: 共有メモリの用意

タイミングデータ送信時には第 6.5.1項で述べた伝送遅延時間の取得のため，タイムスタンプが ConcertinoGetCurrentTime() を用いて記録される．タイムスタンプは拡張 RTP フォーマット (第 6.6節) 内の (unsigned int)ts と，db_entry 構造体配列 (第 6.4節) 内の (unsigned int)timing_send_ts に登録される．

```
extern struct db_entry *db_entry;
(省略)
int timing_send_pkt(struct dvrecv_param *dvrecv_param,
   u_int32_t *buf, int buflen)
{
   int n;
   int i;
   /* クライアントシーケンス番号 0 番から最終番に対して送信 */
   for (i=0; i <= dvrecv_param->client_seq; i++){
      n = sendto(db_entry[i].csock,
         buf, buflen, 0,
         (struct sockaddr *)&db_entry[i].timing_sin,
         sizeof(db_entry[i].timing_sin));

      if (n < 1) {
         perror("timing_send_pkt : sendto");
      }
      db_entry[i].timing_pkt_count++;
   }
   return(1);
}
```

図 6.9: タイミングデータの送信

6.8.3 クライアント：IP ネットワークからのデータ受信

タイミングデータ受信時には第 6.5.2項で述べたクライアント収録環境遅延時間の取得のため，タイムスタンプが ConcertinoGetCurrentTime() を用いて記録される．

6.9 映像・音声データの送受信

6.9.1 クライアント：IEEE1394 からのデータ受信

IEEE1394 デバイスをオープンし，映像・音声データの読み込みを行う．第 6.8節で述べたマスタにおけるタイミングデータ用映像・音声データと同様に，DVTS for MacOSX で用意されている prepare_ieee1394() と main_loop() を用いてデバイスにアクセスする．

6.9.2 クライアント：IP ネットワークへのデータ送信

クライアントはマスタに対して音声のみを二重化した映像データを送信する．これはマスタ側で音声のみのバッファを用意することで，フィードバック送信の効率を高めるためである．

映像・音声データ送信時に送信されるタイムスタンプは，マスタ時間タイムスタンプ，相対

時間タイムスタンプ，クライアント処理時間の 3 つである．

受信したタイミングデータ内拡張 RTP フォーマット (第 6.8節) に記録されたマスタ時間を，そのまま映像・音声データ内拡張 RTP フォーマット内，(unsigned int) ts にコピーを行う．

タイミングデータ内拡張 RTP フォーマット (第 6.8節) の con_pt に演奏開始ビットが立っている場合，クライアントが相対時間の記録を開始する．取得されたタイミングデータは拡張 RTP フォーマット内の (unsigned int)relative_ts に挿入する．

第 6.5.2項で述べた収録環境遅延時間取得の後，拡張 RTP フォーマット (第 6.6節) 内の (unsigned int) client_ts に記録，映像・音声データとともにマスタへと送信する．

6.9.3 マスタ：IP ネットワークからのデータ受信

6.9.4 マスタ：バッファへのデータ書き込み

各クライアントから受信した映像・音声データはクライアント毎に用意した共有メモリに書き込まれる。

6.9.5 マスタ：タイムスタンプ処理

クライアントから受信した映像・音声データ内の拡張 RTP フォーマットのタイムスタンプの読み込み，受信時間の登録を行う．対象となるのは映像・音声データ受信時間，タイミングデータ送信時に記録されたマスタ時間 ((unsigned int) ts)，クライアント内部処理時間 ((unsigned int) client_ts) の 3 つである．

これらの時間は db_entry 構造体配列 (第 6.4) の映像・音声データ受信時間は (unsigned int) data_recv_ts，クライアント内部処理時間は (unsigned int) client_ts に登録される．タイミングデータ送信時に記録されたマスタ時間と受信時間との差である (unsigned int) transmission_ts，transmission_ts よりクライアント内部処理時間を引いたネットワーク伝送遅延 (unsigned int) network_ts に記録される．

6.10 フィードバック送信部・受信部

6.10.1 マスタ：フィードバックデータの送信

フィードバックに用いられる映像は他のクライアントに用意された音声データ用共有メモリから読み込み，送信を行う．フィードバックデータ送信を行う AddConcertinoFeedbackThread() は 1 つのスレッドで行うタイミングデータ送信と違い，クライアント数分の送信スレッドを作成する．このため，これまで述べてきたタイムスタンプによる送信時間調整などが可能である．

6.11 ユーザインターフェースの提供

本実装ではユーザインターフェースを GUI(Graphical User Interface) によって提供する．

クライアントの Concertino Negotiation ボタンを押すことでマスタへの接続要求が送信される．マスタからの映像・音声データ転送用ポート番号を受信後，Start Concertino ボタンを押

すことにより，前述したタイミングデータ受信スレッド，タイミングデータ IEEE1394 出力スレッド，フィードバックデータ受信スレッド，フィードバックデータ IEEE1394 出力スレッド，映像・音声データ読み込み・送信スレッドが開始される．

マスタでは Start ボタンを押すことでクライアントからの音声データ受信・登録スレッドが開始される．登録終了後，Start Concert ボタンを押すことにより，前述したタイミングデータ送信スレッド，及びクライアント数分の映像・音声データ受信スレッド，フィードバックデータ送信スレッドが作成，開始される．

図 6.10: クライアント：スクリーンショット

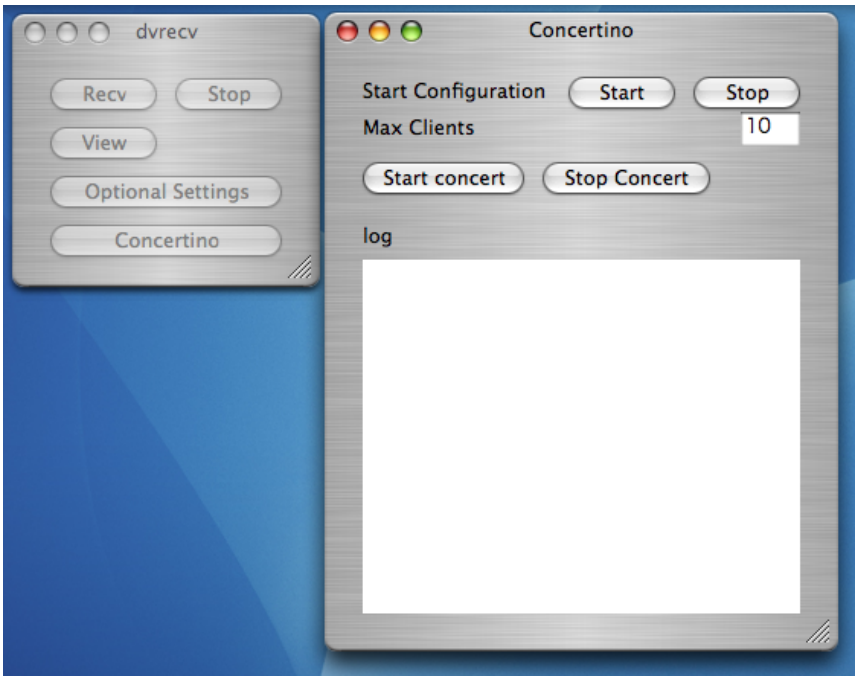


図 6.11: マスタ：スクリーンショット

6.12　まとめ：オーケストラ演奏機構の実装

本章では第5章に基づいて行った実装について述べた．次章では実装された本機構を計測，評価を行う．

第7章　評価：本機構の実現した機能

本章では，第6章において実装された本機構に対する評価について述べる．

7.1　本機構の実現した機能

本システムではフィードバック機能を持ったオーケストラ演奏機構を実現した．満たした機能は以下の4つである．

- 他者の音声データをフィードバックする機構を設けた
- 複数クライアントの管理を行うクライアントデータベースの構築を行った
- 遅延時間管理機構を構築した
- 拡張 RTP フォーマットによる通信を行った

第4章で述べた，既存のネットワークを用いた音楽制作技術のうち，インターネットに対応した坂本龍一 MIDI ライブ，長野オリンピック開会式との比較を定性評価によって行う．
定性評価は以下の4つに対して行った．

- フィードバック機構の有無
  他の演奏者の音声データを取得，和音やダイナミックスの確認の可否
- タイミングの共有
  遠隔地に演奏者が点在した場合，演奏の開始やリズムやテンポを共有する手段の有無
- 有効範囲
  各演奏者の存在する拠点間の距離が大きいほど，有効範囲が高いと判断する
- 映像への対応
  映像に対応することにより，幅広い他分野への応用性

評価結果を図7.1に示す．各項目に対し，それぞれの機構が対応している場合は○，対応していない場合は×で表す．
他の演奏者の演奏を聴取できるフィードバック機構は本機構のみに備わっている．
演奏の開始，リズムを演奏者が取得する機構は，指揮者とオーケストラの映像・音声を送信する長野オリンピック方式と，指揮者映像，及び音声の転送に対応する本機構に備わっている．
有効範囲は本機構のみが改善の余地があるとして△とした．今後，母体である DVTS for MacOSX の受信機構の改善を行うことで有効範囲の拡大が期待できる．

	坂本龍一 MIDI ライブ	長野オリンピック開会式	本機構
フィードバック	×	×	○
タイミングの共有	×	○	○
有効範囲	○	○	△
映像への対応	○	○	○

図 7.1: 本機構と既存技術の比較

映画製作，報道といった他分野への応用の際に重要となる映像への対応は mLAN を除く全ての方式が対応している．

以上の比較より，本機構は他の方式に比べてオーケストラを対象とする音楽共同制作環境に適しており，有効であるといえる．また，映像を用いることによる他分野への汎用性も備えている．

7.2 各データ送信安定性の検証

本節ではクライアントから送信される映像・音声データ，マスタから送信されるタイミングデータ，フィードバックデータの 3 つのデータに対し，パケット送信の安定性と粒度の測定を行う．

測定環境を図 7.2に示す．クライアントはマスタから受信したタイミングデータ，フィードバックデータを IEEE1394 機器に出力する．そのため，クライアント側にはそれぞれ 2 台のメディアコンバータを IEEE1394 ハブを介して接続する．また，映像・音声データの入力を IEEE1394 機器 (DV カメラ) から受ける．

測定に用いた機器を表 7.1に示す．測定はフィードバックデータ送信時の転送機器のパフォーマンスを計測するため，マスタと 2 台のクライアントの合計 3 台の計算機を用いる．

表 7.1: 測定に用いた計算機

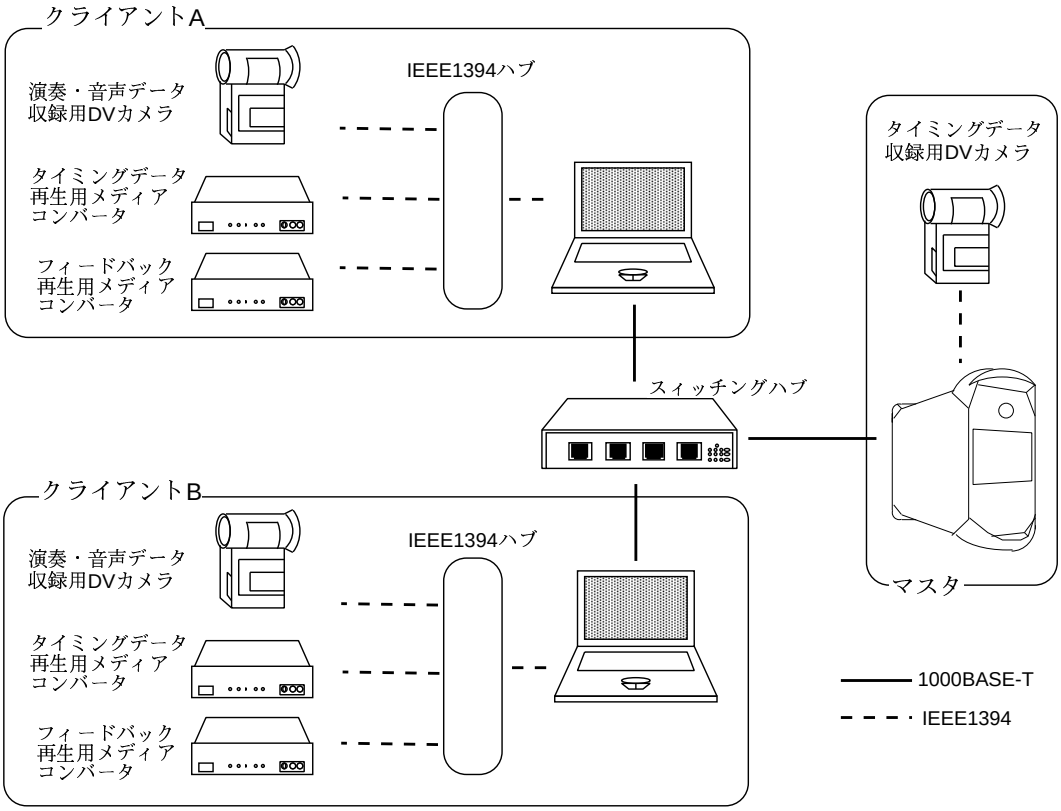
	マスタ	クライアント A	クライアント B
型番	PowerMacG5	PowerMac G4	PowerBook G4
CPU	PowerPC G5 2.0GHz dual	PowerPC G4 1.25Ghz dual	PowerPC G4 800Mhz
メモリ	DDR PC3200 2GB	DDR PC2600 1.5GB	PC133 CL3 SO DIMM 1GB
NIC	1000BASE-T	1000BASE-T	1000BASE-T

7.2.1 クライアント：映像・音声データ送出量の測定

クライアントがタイミングデータ，フィードバックデータ受信を同時に平行して行った際の映像・音声データ送出量を計測した．計測はマスタ側で tcpdump を用いて行う．計算能力の高いクライアント A の映像・音声データを受信するポート番号に対し，10 分間測定した．

以上の測定の結果を図 7.3に示す．クライアントがタイミングデータ，フィードバックデータ

図 7.2: 測定時の環境



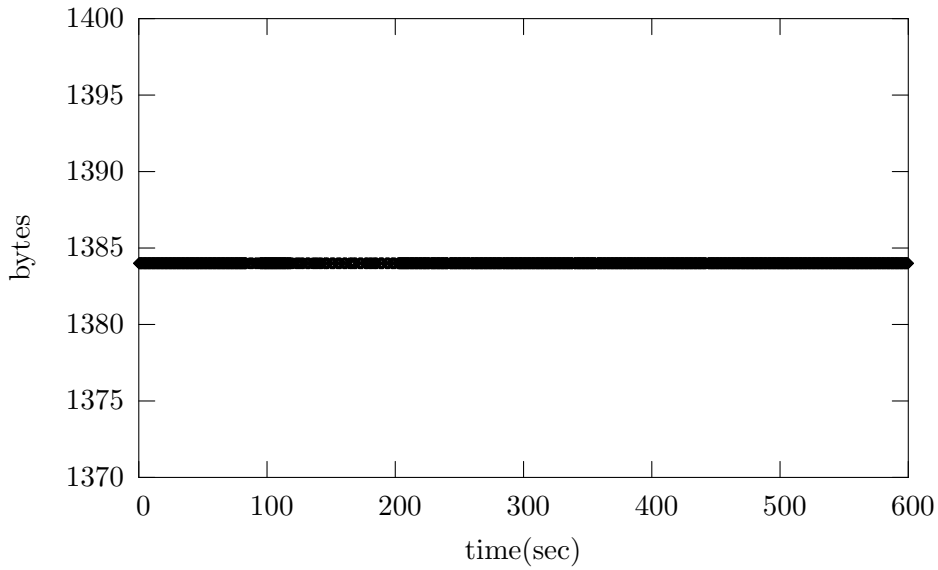


図 7.3: クライアントにおける映像・音声データ送信量

の受信を平行して行いながら一定の映像・音声データを送出していることが分かる.

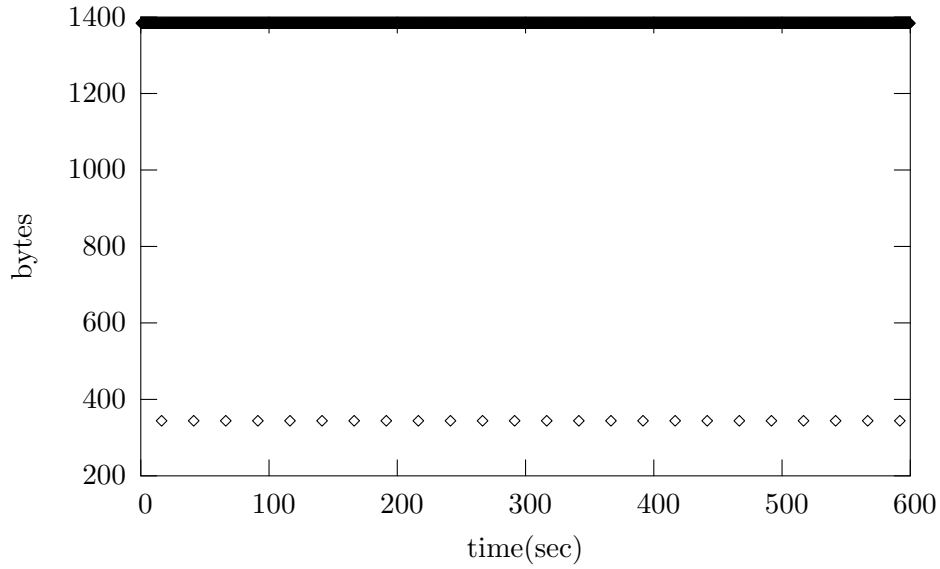


図 7.4: マスタ：タイミングデータ送信量

7.2.2 マスタ：タイミングデータ送出量の測定

マスタが 2 台のクライアントに対して同時にタイミングデータ，フィードバックデータ送信，映像・音声データ受信を平行して行った際の，タイミングデータ送出量を計測した．実験環境，機材は第 7.2節と同様である．計測は計算能力の高いクライアント A において，tcpdump を用い，タイミングデータを受信するポート番号に対し，10 分間測定した．

以上の測定結果を図 7.4に示す．タイミングデータの送信は 1 つのスレッドで全てのクライアントに対して行われる．そのため若干のばらつきが発生するが，ほぼ一定の映像・音声データを送出していることが分かる．

7.2.3 マスタ：フィードバックデータ

マスタが 2 台のクライアントに対して同時にタイミングデータ，フィードバックデータ送信，映像・音声データ受信を平行して行った際の，フィードバックデータ送出量を計測した．実験環境，機材は第 7.2節と同様である．計測は計算能力の高いクライアント A において，tcpdump を用い，フィードバックデータを受信するポート番号に対し，10 分間測定した．

以上の測定結果を図 7.5に示す．フィードバックは他のデータとは違い，音声データのみを対象として行うためデータ送出量が小さい．マスタがタイミングデータ送信，映像・音声データ受信を平行して行いながら一定のフィードバックデータを送信していることが分かる．

7.3 演奏可能許容時間と処理速度

演奏開始から他の演奏者の音声を聴取するまでの時間が，80ms 以内に行われる必要がある．本節では演奏の開始の合図となるタイミングデータ受信時から，映像・音声データの送信，フィードバックの受信までの一連の処理に必要な時間を計測した．80ms から計測された値の差分を

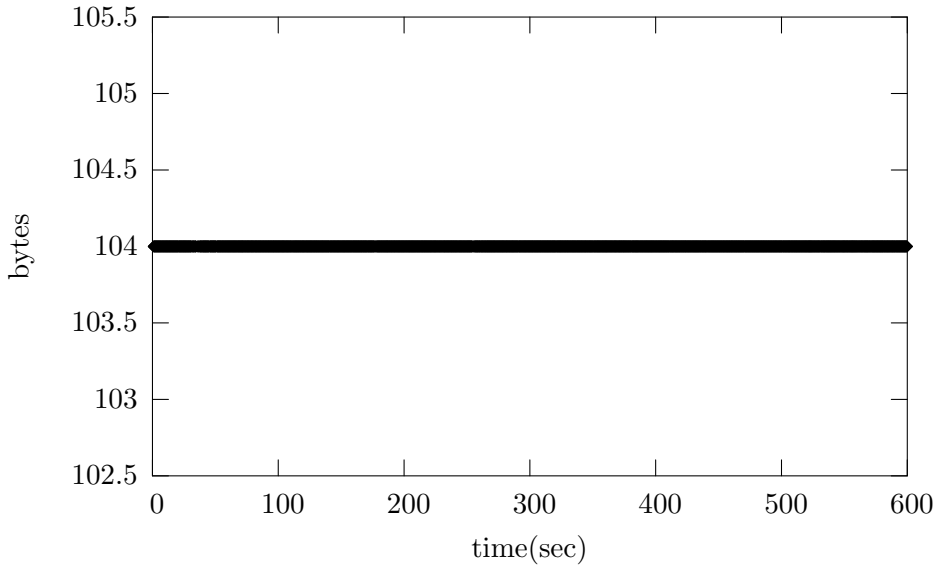


図 7.5: マスタ：フィードバックデータ送信量

取ることにより，ネットワーク伝送時間として扱える時間が導きだされるため，本研究の有効範囲が求められる．

演奏者が演奏を開始し，他の演奏者の演奏を聴くまでの時間を拡張 RTP フォーマットの一部を変更することにより計測を行った．実験環境，機材は第 7.2節と同様である．計測は計算能力の高いクライアント A において，10 分間行った．

演奏開始をタイミングデータ受信時の時間と仮定した計測を行う．マスタがタイミングデータを 1 から t まで送信する際にパケット *(t)* 内の拡張 *RTP* フォーマット内にマスタ側 *CPU* 時間 $M_{(t)}$ を挿入する．この際の単位は *10msec* である．クライアントがタイミングデータ t 受信時に，取得されるクライアント側 *CPU* 時間を $C_{(t)}$ とする．受信したクライアントは，マスタ側 *CPU* 時間 $M_{(t)}$ を映像・音声データ内の拡張 *RTP* フォーマットに複写する．映像・音声データを受信したマスタは，映像・音声データ内のマスタ側 *CPU* 時間 $M_{(t)}$ をフィードバックデータ内 *RTP* フォーマットに複写，クライアントに転送する．クライアントが拡張 *RTP* ヘッダ内に $M_{(t)}$ を含むフィードバックデータ受信時の *CPU* 時間を $F_{(t)}$ とする．この際，タイミングデータ受信からフィードバック受信するまでの時間 $D_{(t)}$ は

$$D_{(t)} = F_{(t)} - C_{(t)}$$

によって求められる．

以上の測定の結果を図 *7.6*に示す．数 *ms* の遅延時間で処理を行う場合が多いが，ばらつきが目立つ．平均所要時間は *37ms* であるため，実験環境におけるネットワーク伝送遅延を *0* と仮定した場合，ネットワーク伝送遅延が往復で *43ms* 以内の拠点であれば演奏可能であると言える．

所要時間のばらつきは本実装の母体となっている *DVTS for MacOSX* に起因するところが大きい．*DVTS for MacOSX* の受信機構のパフォーマンスの改善を行うことにより，本研究の有効範囲の拡大が予想される．

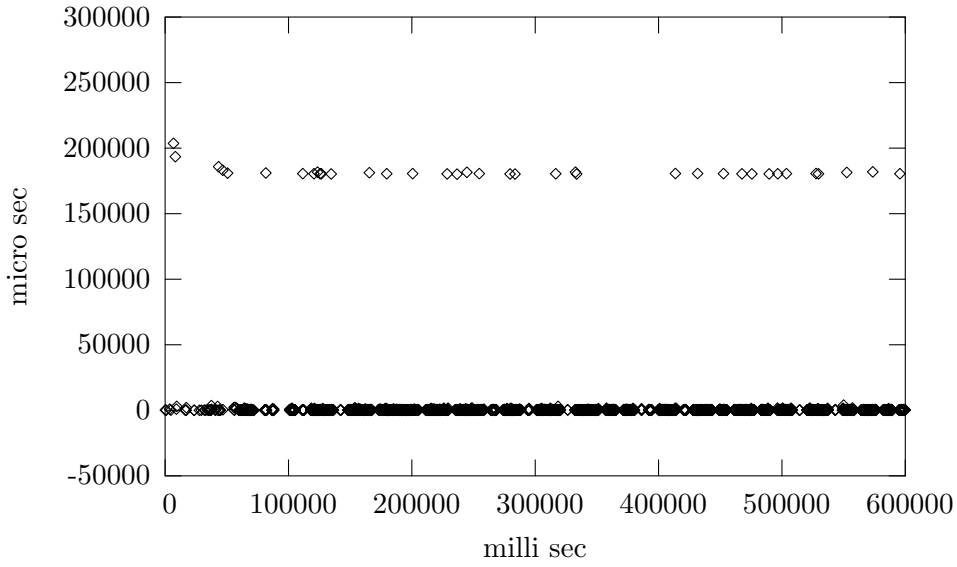


図 7.6: 演奏可能許容時間と処理速度

7.4 考察

複数のクライアントに対し，安定したフィードバックデータ送信を実現することができた．またクライアントでは，タイミングデータ，フィードバックデータの受信とともに安定した映像・音声データ送信を実現することができた．

タイミングデータ送信の際に若干のばらつきが見られる．実際の用途や環境，転送機器の仕様によってタイミングデータのクォリティに対する要求は異なる．クライアント毎にタイミングデータ処理を分割することで品質向上の可能性があるが，マスタの負担増加に伴う映像・音声データへの影響，スケーラビリティの問題がある．タイミングデータに関しては複数クライアントへの送信を単一スレッドで行うパフォーマンス重視のモードと，個別のスレッドで行うクォリティ重視のモードをユーザインタフェースから選択できることが望ましい．

演奏の開始からフィードバック受信までの所用時間にはばらつきが存在する．現在の本機構の有効範囲はネットワーク伝送遅延が往復で *43ms* 以内に限定される．しかし所用時間が数 *ms* の範囲に集中している値が多いことから，受信機構の改善により本機構の有効範囲が拡大する可能性が高い．

7.5 まとめ：本機構の実現した機能

本章では第 *6*章で述べた実装に基づいた実装を計測，評価を行った．その結果，安定したタイミングデータ，フィードバックデータ，映像・音声データの送受信が可能な音楽共同制作機構を実現した．

次章では本機構の発展について述べる．

第8章　結論：オーケストラ演奏機構

第 6章で本研究の実装に関して述べた．本章では第 6章から導き出された内容より考察，まとめを行う．

8.1　まとめ

本研究ではフィードバックを用いたオーケストラ演奏機構の構築を行った．本機構を用いることにより，指揮者映像，他の演奏者の音声を参照しながら演奏を行うことができる．演奏の基準となる映像データの転送，他の演奏者の音声のフィードバックを行うことにより，映像データによる演奏タイミングの共有と，音声データによる和音・ダイナミックス確認の両立ができる．

実空間の音楽共同制作の分析を，コンサートホールを対象に行った．コンサートホールにおける演奏では，壁や天井からの反響時間が長く，ステージに音が返ってくる際には遅延が発生する．特にステージからの距離が離れている演奏者の音声は遅延して到達するため，演奏のタイミングは指揮者などを視覚によって確認する必要がある．また，視聴者にはコンサートホールの設計によって反響音を操作し，演奏者間の同期が確保された状態で届く．

演奏者，指揮者，視聴者がネットワーク上に分散した環境において音楽の共同制作を行う場合，必要な事項は *1)* 演奏者が他の演奏者の音声聴取 *2)* 演奏者間のタイミング，リズム共有 *3)* 視聴者は全ての演奏者の演奏を同期された状態で視聴，の *3*つである．

ネットワークを経由し，音楽の共同制作を行う機構の例として坂本龍一 *MIDI* ライブと長野オリンピック開会式がある．この二つに対し検証を行った．検証より本研究における必要要件のうち，遅延時間の管理はクライアントによるバッファリング *(*坂本龍一 *MIDI* ライブ方式*)*，タイムラグアジャスタの使用 *(*長野オリンピック開会式方式*)* などの手法を用いて実現されている．しかし，他の演奏者の音声聴取，指揮者映像などの送信によるタイミングの共有の二つを同時に満たしては居ない．

以上の問題点を解決するために，以下の *4*つのモジュールの設計を行った．

- タイミングデータ送受信部
  指揮者映像などを送受信することにより，演奏のタイミングの共有を行う
- フィードバックデータ送受信部
  他の演奏者の音声をマスタからクライアントに対し送信することで音声データからの和音，ダイナミックスの確認を行う．
- クライアントデータベース
  クライアント毎に異なる *IP* アドレス，音声データ送信用ポート番号といった状態の他，映像・音声データ用バッファなどを登録する

- 遅延管理部
  ネットワーク伝送遅延時間，クライアント内部処理時間の取得を行う．

IEEE1394 から *Isochronous* 転送を行い，安定した映像・音声データの入出力を行う．これらの映像・音声データをネットワークを介して転送を行うために *DVTS*，特に *DVTS for MacOSX* をベースとした実装を行った．

本研究により，ネットワークを介した音楽共同制作が可能となった．次章では本研究の発展について述べる．

第9章　本機構の発展

本章では本機構の実用性，汎用性を高める上で必要な事項を述べる．

9.1　今後の展望

本研究における今後の発展は以下の事項が挙げられる．

- 操作性の向上
  本機構では最小限の *GUI* を実装するに留まっている．クライアント毎のフィードバック伝送時間の送信時間設定，プライオリティ設定による特定の演奏パートのデータ保護など，様々な設定が考えられる．これらの設定は一般ユーザにとって使い易い *GUI* によってなされる必要がある．よりアプリケーションに柔軟な *GUI* を適用することができる *Cocoa API* による実装が必要となる．

- *Viewer* の採用
  本機構ではクライアントにおいてタイミングデータ受信，映像・音声データ送信，フィードバック受信の *3* 箇所で *IEEE1394* への書き出しを行う必要がある．そのため，*3* 台の *IEEE1394* 機器が必要となるため，ユーザへの負担が大きい．受信したデータを *PC* のディスプレイ上で再生する *Viewer* の作成により，*IEEE1394* 機器は映像・音声データ送信用の *1* 台のみを使用することになり，簡易なシステム構築ができる．

- スケーラビリティの向上
  本機構ではタイミングデータ，及びクライアントの映像・音声データに対するフレーム間引は行っていない．クライアントはタイミングデータ受信，映像・音声データ送信，フィードバック送信の *3* つのストリームに対する処理を行う必要がある．またマスタは *3* つのストリームをクライアント数分処理する必要がある．映像・音声データのフォーマットを含め，処理の効率化を行い，スケーラビリティの向上を図る必要がある．

- バッファリングパフォーマンスの向上
  本機構はクライアント毎に複数のスレッドを作成，バッファリング，映像・音声データの送受信を行うため，マスタへの負担が大きい．各処理の最適化を図る必要がある．特にバッファリングはクライアント毎に映像・音声データ処理を行うことに加え，タイムスタンプと合わせた処理を行う必要があるため，マスタ *PC* への負荷が大きい．バッファリングアルゴリズムの比較，最適な実装を行うことにより，より効率の良いシステム構築ができる．

- ソフトウェアミキサの作成
  本機構ではクライアント毎に独立したバッファリングを行っている．これらのバッファか

ら映像・音声データを抽出，音声のレベル合わせなどを行った上で出力を行わなければならない．そのためストリーミングに対応したソフトウェアミキサの作成が必要となる．

- 実際のオーケストラでの使用
- *mLAN[12]* との連携
  IEEE1394 デバイスを用い，*Isochronous* 転送によって入出力を行う *mLAN* に対応することにより，応用性の高いシステムの構築を目指す．
- *IEEE1394* デバイス以外への対応
  S/PDIF(Sony/Philips Digital Interface)[29] などに対応することにより，既存の収録環境を活かした汎用性の高いシステムの構築を目指す．*S/PDIF* は出力される音声データに時間軸の概念がないため，*IEEE1394* デバイスに見られる時間軸を追加する必要がある．

謝辞

本研究を進めるにあたり，御指導を頂きました，慶應義塾大学環境情報学部教授村井純博士，徳田英幸博士，同学部助教授の楠本博之博士，中村修博士，同学部専任講師の南政樹氏，重近範行博士に感謝致します．

絶えずご指導とご助言を頂きました独立行政法人通信総合研究所 杉浦一徳博士，*SFC* 研究所 小川浩司氏に感謝致します．評価の過程でご助言を頂きました *WIDE Project 10g-wg* の皆様に感謝いたします．

また，実装時にご助言を頂きました入野仁志氏を始めとする *STREAM KG, DVTS Project* の皆様に感謝致します．

論文執筆に際してお手伝い頂いた慶応義塾大学政策・メディア研究科の入野仁志氏，小柴晋氏に感謝致します．

お手伝い頂いた慶応義塾大学環境情報学部　千代佑氏，松園和久氏，小椋康平氏に感謝致します．来年は君たちです．頑張りましょう．

最後にここまでやる気にさせてくれた *Macintosh, MacOSX* の開発者達に敬意を表します．

参考文献

[1] *Institute of Electrical and Electronics Engineeers, Inc. IEEE Standard-Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Physical Layer Parameters and Specification-for 1000 Mb/s Operation over 4-pair of Category 5 Balanced Copper Cabling, Type 1000 BASE-T.* http://standards.ieee.org/reading/ieee/std/lanman/restricted/8 02.3ab-1 999.pdf, *2000.*

[2] *Institute of Electrical and Electronics Engineeers, Inc. Carrier Sense Multiple Access with Collision Detection(CSMA/CD) Access Method and Physical Layer Specifications -Media Access Control(MA C) Parameters, Physical Layer and Management Parameters for 10Gb/s Operation.* http://standards.ieee.org/reading/ieee/std/lanman/restricted/8 02.3ae-2002.pdf, *2002.*

[3] *digital cinema consortium.* デジタルシネマ研究コンソーシアム *WWW page.* http://dcc.imgl.sfc.keio.ac.jp.

[4] 松原 慶祐. *Effects of auditory feedback on musical instrument.* http:www.jaist.ac.jp/ ksuke-m/research/eafb.html, *2001.*

[5] *Real Networks Corporation. Real Networks WWW page.* http://www.real.com.

[6] *J. Postel.* Transmission Control Protocol, *September 1981. RFC 793.*

[7] *J. Postel.* User Datagram Protocol, *August 1980. RFC 768.*

[8] *Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson.* RTP: A Transport Protocol for Real-Time Applications, *January 1996. RFC 1889.*

[9] ヤマハ株式会社. *Midlive.* http://www.yamaha.co.jp/xg/entertainment/midlive/index.html, *1998.*

[10] *ACOUSTIC REINFORCEMENT TECHNOLOGY INC. The 18th Winter Olympic Games in NAGANO 1998 Audio Design for th e Opening and Closing Ceremonies.* http://www.arte-inc.com/ardbf/olympic98/index_e.htm.

[11] IEEE Standard for a High Performance Serial Bus, *1995. IEEE computer society.*

[12] *YAMAHA Corporation. mLan Alliance WWW page.* http://www.yamaha.co.jp/tech/1394mLAN/.

[13] Yamaha Corporation. Proposal for Audio and Music Protocol Draft Version 0.32. Aug. 1996.

[14] Yamaha Corporation. Specification for Audio and Music Data Transmission Working Draft Version 0.90f3. Jul. 1997.

[15] P.Winsor and G.Delisa. Computer music in C. 1987. ISBN:0-8306-3637-4.

[16] ヤマハ株式会社. ヤマハインターネット MIDI ライブ 『Mid Live Vol.3』一般公開実験, *April 1997.*

[17] NTT Corporation. TwinVQ WWW page. http://www.twinvq.org/.

[18] Yamaha Corporation. TwinVQ WWW page. http://www.yamaha.co.jp/xg/SoundVQ/.

[19] ヤマハ株式会社. ヤマハ『インターネット MIDI ライブシステム』公開実験, *April 1997.*

[20] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394-1995 High Performance Serial Bus. Aug 1996.

[21] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394a-2000, IEEE Standard for a High Performance Seri al Bus,, Amendment 1. http://standards.ieee.org/reading/ieee/std/busarch/1394a-2000 .pdf, *Jun 2000.*

[22] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394b-2002, Amendment to IEEE Std 1394-1995. http://standards.ieee.org/reading/ieee/std/busarch/1394b-2002 .pdf, *Dec 2002.*

[23] NHK 放送技術研究所. *NHK* 技研だより 技術動向 長野オリンピックの新機材. *Arg. 1998.*

[24] D.L. Mills. Network Time Protocol (NTP), *September 1985. RFC 958.*

[25] A.Ogawa. DVTS (Digital Video Transport System) WWW page, *November 2001. URL:http://www.sfc.wide.ad.jp/DVTS/.*

[26] 小川 晃通. 協調的輻輳制御を用いた映像配信機構の設計と実装. *1999.*

[27] Tsuyoshi Hisamatsu, Akimichi Ogawa, Sugiura Kazunori, Osamu Nak amura, Jun Murai. Software compatibility and human interface for dv over ip. Symposium on Applications and the Internet Workshops, *pages 188–191, Jan 2003.*

[28] Apple - Mac OS X, *2002. URL http://www.apple.com/macosx.*

[29] IEC 60958 Digital audio interface. http://www.iec.ch, *Dec 1999.*