音楽の遠隔擬似ライブ録音システムの試作

2003MT043 近藤 美希　　2003MT073 中野 好絵
指導教員　後藤 邦夫

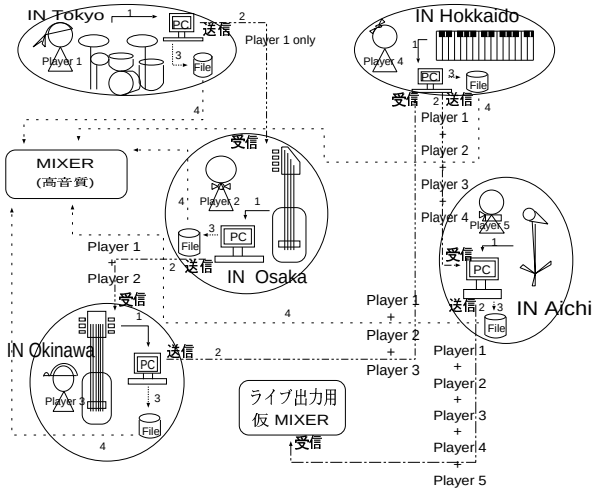
1　はじめに

現在，ネットワークを通して音楽の販売・試聴が可能となっているが，演奏データをリアルタイムで送受信できる遠隔録音システムは実現していない [3] ．これは送受信時に起こる遅延やパケットロスが原因である．

日々インターネットを用いた音楽利用の需要は増えており，今後もさらに普及していくと考えられる．このような中で，離れた場所にいても音楽を共有できるといいのではないかと考え，遠隔録音システムを試作する．

本研究では，複数人でネットワークを介し，リアルタイムで演奏データを送受信し，違う場所にいてもそれぞれの音楽を聞きながら演奏できるシステムを作成する．また，演奏するだけでなく，それぞれの演奏をミックスダウンする機能を付け加える．

演奏形態は，楽器別で演奏するマルチトラック型の演奏方法を使用する．演奏順序は基本となるリズム楽器から始める (図 1)．この他にはコンダクタ型の方法 [2] がある．1 パート毎に演奏するため， ポピュラー音楽の演奏には適しているが，全パート同時に演奏するクラッシック音楽にはあまり適さない．



矢印の説明

1：自らの演奏をPCに取り込む(次の演奏者への送信用・PCへの録音用)

2：次の演奏者のPCへ演奏データを送信する

3：データサイズを軽減していない演奏をファイルに保存する

4：作成したファイルをMIXERへ送信する

図 1　遠隔ライブ疑似録音システムの概要

データの送受信時には遅延やパケットロス，パケット入れ替わりが生じることがあるので，この補正処理を作成する．ネットワークエミュレータでこれらの障害を模倣し，どの程度のパケットロスに対応できるかを考察することにより，システムの性能を検証する．

システム構築・考案は共同で行い，その中でも特に中野はネットワーク構築を，近藤は機材調整を担当した．

2　システムの概要

ここでは本研究で作成するシステムの概要を記述する．

インターネットを用いたライブ録音システムでは，遅延やネットワークバンド幅を考慮する必要がある．インターネットでは数十ミリ〜数百ミリ秒遅れが発生すると考えられる．

一般に家庭で使用できるバンド幅は 500kbps〜数 Mbps である．このバンド幅を越えないようなデータ量の通信を考える必要があるため，データを圧縮する．データの圧縮法については 3 節で詳述する．

2.1　マルチトラック録音

本研究では各トラックを別々に保存していくマルチトラック録音方法を用い，曲を作成する．この方式では，ミキシング時に各演奏の音量やエフェクトなどの調節が可能となり，楽曲にした時の完成度が高くなる．マルチトラック録音以外の方法として，一つのトラックに音を重ねていき，録音していくものがある．しかしこの方法では録音時にそれぞれの演奏の同期を取るのが困難であり，それぞれの演奏の音の調整も録音後には不可能である．

マルチトラックレコーディングに用いられる市販・家庭用の HDD レコーダーは 16 トラック以上のものが一般的であり，本研究でもこのトラック数に近づける．また，送信時には同期を取りやすくするために，1 パケットに全トラックの音を入れ，送信する．

2.2　提案するシステム

本研究ではリアルタイム性を重視する点から UDP 通信方式を取る．通常のインターネットの MTU は 1500 であり，IP ヘッダや UDP ヘッダを除くとデータは約 1470 バイト送ることが可能である．大半の一般家庭では ADSL を利用しており，PPPoE を用いる．PPPoE では送信可能なデータはインターネット通信よりも少なくなる．この時の UDP ペイロードは余裕を持たせ，1400 程度であると考える．16 トラックに近いトラック数をこの UDP ペイロード以内で送信したいため，トラック数を 14，1 トラックを 100byte に設定する．

データサイズを表すものにサンプリング周波数と量子化ビット数がある．マルチトラック HDD レコーダでは 16bit 以上・48kHz，音楽 CD では 16bit・44.1kHz

のものが一般的であり，音楽 CD の音質と同等であれば演奏に支障はないと考え，本研究では音楽 CD と同等の設定を用いる．この時の必要最大バンド幅は 16 × 44100×14=9.8784(Mbps) である．これでは，一般回線のバンド幅を超えてしまうので，データを圧縮する．使用バンド幅を抑えるため，16bit を G721 ADPCM で 4 ビットに圧縮して $\frac{1}{4}$ にする．この場合，最大で約 2Mbps となる．さらにサンプリング周波数を半分の 22.05kHz にするオプションを設定する。この設定は 4 節で説明する設定ファイルにおいて選択する．この設定を行うとバンド幅は $\frac{1}{8}$ になり，最大で約 1Mbps 使用する．ただし，この計算で得られた必要バンド幅は IP ヘッダや UDP ヘッダなどが含まれていないため，実際の必要バンド幅は計算値よりも少し大きくなる．

次に演奏データのモードについて説明する．1 つのサウンド装置において，録音と再生のモードは同一にしか設定できない．再生はステレオが望ましいので，録音もステレオにする．量子化ビット数・サンプリング周波数も含め，全演奏者は同一の音声形式を用いる．

本研究では，遠隔地からの演奏をインターネットを用いて他の演奏者へ送信し，受信した演奏データに合わせた演奏が可能なシステムを考える (図 1)．

最初の送信者は送信のみを，2 人目以降は送信・受信を同時に行う．演奏スタートと同時に演奏している音を次の演奏者に送信する．次の演奏者は演奏の受信と同時に再生し，これに合わせて演奏を始め，同時に演奏データと受信した音を共に次の演奏者へ送信する．この作業を最終演奏者 (Player 5) まで繰り返し，最終演奏者は今までの演奏を全て仮 MIXER に送信する．MIXER では，各自で録音したファイルをミックスダウンし，楽曲を作る．このミキシングの作業にはフリーソフトウェアである Ecasound[4] を用いればよい．

3 データ形式

この節では送受信するデータの形式について述べる．

3.1 データ圧縮

本研究ではバンド幅を考慮し，演奏データを圧縮して送信する．データ圧縮には可逆圧縮と非可逆圧縮の二種類がある．非可逆圧縮方法では元に戻すことはできないが，人の感覚に伝わりにくい部分を減らしているため，データ欠落は目立たない．非可逆圧縮は可逆圧縮に比べ，圧倒的な圧縮率を得ることができる．したがって本研究では非可逆圧縮を用い，その中でも PHS などに用いられている G721 ADPCM 方式を用いる．

ADPCM とは音声を符号化する手法の 1 つで，現在の信号とその前置周波数との差分を符号化し，圧縮する．単純な符号化なので，処理時間が短くリアルタイムに適している．データ受信後はデコーダを通した後に，データを再生させる (図 2)．圧縮・デコーダ・ダウンサンプリングはプログラム内で実現する．

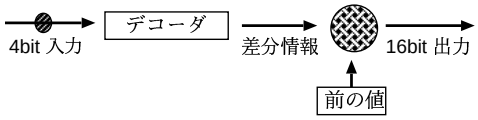


図 2　ADPCM のデコーダ

3.2 送信ヘッダについて

本研究では送信データをプログラム内で構造体にする．送信データはヘッダ部と演奏データ部に分けられる．送信ヘッダの中にはプロトコルバージョン・チャネル数・パケット番号を格納し，次の演奏者に送信する (図 3)．

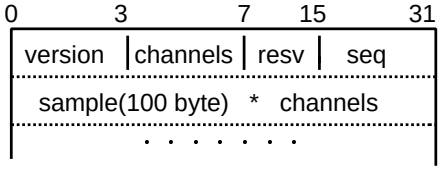


図 3　構造体のヘッダの中身

sample には，各トラックの演奏データを 100byte ずつ入れていく．sample は，構造体に入れヘッダと共に次の演奏者に送信する．パケットロスや入れ替わりを検出，補正するためにヘッダの中にパケット番号 (seq) を加える．そして再生時には受け取ったデータのチャネル数が必要なため，channels も含む．このチャネル数は送信する度に増えていく．ヘッダのサイズを調整するために，resv(未使用) を用意する．ヘッダ情報のバイト数は演奏データの量子化ビット数に合わせ，バージョン・チャネル数が unsigned int 型の 4bit，パケット番号を unsigned short 型の 16bit にした．

4 システムの実現

本研究では，遠隔でのライブ録音を実現する．マルチスレッドを用いて送受信を同時にすることにより，リアルタイムでの演奏を可能にした．

4.1 設定ファイル

演奏者の役割・楽器名・トラック数・受信先アドレス・受信ポート番号・送信先アドレス・送信ポート番号・保存ファイル・再生ファイルを示すプログラム実行設定ファイルを各演奏者毎に作成する (図 4)．これは，コマンドラインの複雑をさけるために作成する．

MyRole では，第 1 演奏者，第 2 演奏者以降，MIXER のいずれかを指定する．Instrument では，各自演奏する楽器名を示す．Meteronome の場合は，プログラム内でメトロノーム音を自動生成する．PrevHost，PrevPort は，演奏データを受信するさいに指定する．NextHost，NextPort は，演奏データを送信するさいに指定する．RawOut は，演奏データをファイル保存する時に用い

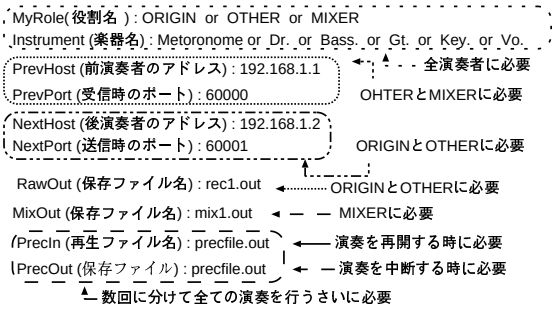


図 4　設定ファイルの中身

る．MixOut は仮 MIXER でデータをミックスアウトするさいに用いる．ネットワークに送信するパケット形式でファイル入出力する時は，PrecIn, PrecOut を指定する．これを用いると，ハードディスクのみを使用した多重録音，または，ネットワーク経由で数回にわけて演奏・録音できる．

4.2　スレッド処理

本研究では，2 個のスレッドを同時に動作させ，送受信用にソケットを 2 個用意した．

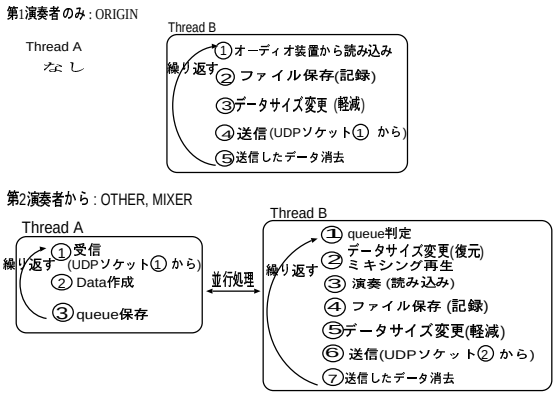


図 5　スレッド処理の流れ

第 1 演奏者は送信のみなので，スレッドは 1 個でよい．第 2 演奏者からは送受信するので，スレッドが 2 個必要である (図 5)．ORIGIN では，演奏データを読み込み RawOut に書き込む．そしてデータサイズを軽減し，次の演奏者にデータを送信する．OTHER・MIXER では，前の演奏者からデータを受信し，queue を作成・データを保存する．これらの作業を 1 つ目のスレッドで繰り返す．また受信後データの入れ替わり・ロス判定，データ再生，演奏データ保存，サイズ軽減，データ送信を 2 つ目のスレッドで繰り返す．

4.3　パケットの処理

演奏データの受信時に遅延・ロス・入れ替わりなどが発生する．そのまま再生すると，演奏の途切れ・音のひずみなどが考えられる．音切れ，ひずみなどを軽減するためパケット処理を考えた．

- データ受信時に発生するパケットの遅れ
  データ受信時にパケットの遅れが発生すると，音が途切れる．音切れを発生させないため，受信データをバッファにためる機能を考えた．バッファ queue に一定量データを保存し，再生させる．
- 入れ替わりについて
  データを queue に保存するさいに，送信データ順に届いているか確かめる．前に到着したデータと今到着したデータを比較し，データを送信した順に受信しているか確認する．入れ替わりがある場合はデータ順序を入れ替え，queue に保存する．
- データロスの考慮
  queue からデータを出力するさいにロス分のデータを復元する．直前に queue から出力したデータと今出力するデータを比較し，番号が続いていなければロスが発生している．ロス発生時に再生するデータは，直前に出力したデータをコピーし使用する．

seq は unsigned short 型で定義しているため，0〜65535 までになるので，65535 以上の数を表すことが出来ない．送信データが 65537 個目の場合，seq は 0 になる．受信した seq の差が，$\frac{65536}{2}$ 以上である時，0 に戻ったと判別する．データ入れ替え・ロスだと判断しないよう seq に 65536 を加えた値を格納する int を用意する．

5　実験/評価

ここでは作成したプログラムを用いた実験と，そのさいに得られたいくつかの結果を比較し，システムの性能を評価する．

5.1　実験環境

実験は学校内で擬似ネットワークを作成し，複数の演奏を同時に行う．実ネットワークを模倣するため，遅延・ロスを起こすエミュレータ GINE2R0.7[1] を使用する．実験でのネットワーク構成を図 6 に示す．

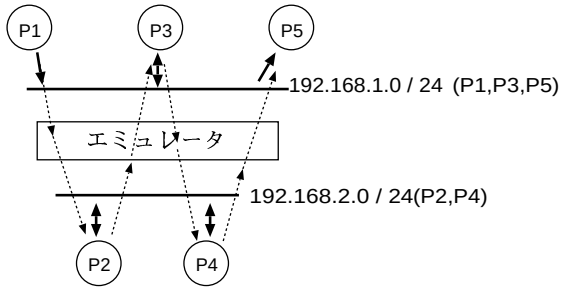


図 6　実験ネットワークを表した図

実際のネットワークではどの程度の入れ替わり・ロスが生じるのかを調べるため，学校－中野自宅間で通信をした．この結果，実際のネットワークではほとんど入れ替わりやロスは発生しなかった．今回は 2 点間のポップ

数が 10 であったが，距離が遠くなり，ポップ数が増えるとこれらが発生する可能性があるため，エミュレータではロス率や入れ替わりを小さな値に設定する．

5.2 実験

実験ではノート PC2 台，デスクトップ PC2 台，合計 4 台を使用した．演奏に使用した楽器はドラムマシン，ベース，ギター，キーボード，マイクの 5 つである．デスクトップ 1 台はエミュレータとして用いた．5 パートを 3 台の PC で演奏するため，演奏を 2 回に分けた．

ノート PC では通常モノラル録音しかできないので，オーディオキャプチャを用いて音を取り込んだ．本研究では UA-3FX・UA-4FX(Roland 社製品) を使用する．

5.3 結果

今回の実験では入れ替わり量を変えず，ロス率を 1%〜10%，演奏のスピードを早いものと遅いもので実験した．結果は以下のとおりである (表 1，表 2)．

表 1　ロス率による比較

ロス率	リズム	音質	その他
1%〜3%	乱れない	若干途切れる	音停止有り
4%〜7%	若干乱れる	ひずみ有り	音停止有り
9，10%	乱れる	悪い	音停止有り

表 2　演奏速度による比較

演奏速度	リズム	音質	その他
♪ 130	乱れない	良い	特になし
♪ 80	各演奏のズレが目立つ	良い	特になし

また，作成したシステムの性能を評価するために，パケットロスや入れ替わりの処理を入れずに実験したところ，リズムが狂い，音楽と認識することが難しかった．したがって，処理によって音が改善できたといえる．

5.4 評価

実験結果より，ロス率が 1，2% 程度であれば問題無くシステムを実行できることがわかったが，ロス率が 4% 以上になるとパケットの補完が増え，音質が悪くなった．さらに 9% 以上になるとリズムも乱れ，演奏が困難になった．大きなロスや入れ替わりが生じた場合，前のデータをコピーして再生させるため，ADPCM で再生させた時に前との音の差が実際の音と異なるため，音が崩れてしまう．

また，演奏のずれが起きる原因としては，オーディオ装置の読み込み時間とネットワーク遅延が考えられる．本研究で用いたノート PC ではオーディオ装置の機種は ALi M5451，ALSA バージョンが 106 を使用している．ネットワーク遅延を変化させてもずれに変化がなかったので，オーディオ装置の原因であることが分かった．ネットワーク遅延を大きくしてもシステムが正常に稼働したことから，国内外でのシステム利用が可能であるといえる．そこで usb オーディオキャプチャを介した場合と介さなかった場合の実験をした．その結果，オーディオキャプチャを介さなかった場合にずれが生じたので，特にノート PC 内蔵のオーディオ装置の書き込みに問題があるという結論に至った．さらに，ネットワークからの入力・出力時間がずれの原因であるとも考えられるため，擬似ネットワークに繋がずに実験した．この結果，ずれが発生したので，ネットワークからの入出力ではないことも判明した．

演奏停止の原因を調べるため，エミュレータを介した場合と介さなかった場合を実験した．その結果，エミュレータを介さなかった場合では演奏停止は起こらなかったため，エミュレータが原因であることがわかった．

演奏データの受信後，一定量バッファにためた後に再生するため，同時に演奏できない．パケット順序を正確に並び替えるため，バッファに溜めるデータ量を多くしたが，これが逆にリアルタイム性を欠く結果になった．

6 おわりに

本研究により，UDP を使ってデータを送受信することでロスを少なくし，入れ替わりやロス処理をすることで，音として最高のものを送受信することが可能となる．しかし，大きなロスや大幅な入れ替わりによって生じるパケット損失において，音の補完が十分にできない．この圧縮法に適した補償方法を検討する必要がある．

演奏データのずれは複数人で行うライブにとって重要な問題である．この処理はまだ実現できていない．書き込み速度のずれを修正することは難しいので，ずれを見越した再生方法を考えなければならない．さらに，リバーブの完成によってライブ感が増すと予想される．

また，作成したシステムはコマンド入力による実行であるので，操作性が十分でない．CUI から GUI に移行することで，さらなる利便性が得られると予想される．

参考文献

[1] Ihara, A., Murase, S. and Goto, K.: IPv4/v6 Network Emulator using Divert Socket, *Proc. of 18th International Conference on Systems Engineering (ICSE2006)*, Coventry, UK, pp. 159–166 (Sep. 2006).

[2] Nakamura, O., Sugiura, K., Yamamoto, S., Shigechika, N., Kato, A., Hasebe, K. and Murai, J.: Internet Metoronome: An Experimental Remote Jazz Jam Session with Uncompressed HDTV Transmission over Lightpath, *IECE TRANS. COMMUN.,VOL89-B,NO.4*, pp. 1052–1058 (April. 2006).

[3] Online Recording Community: digital musician net (accessed April 2006). http://www.digitalmusician.net/.

[4] Vehmanen, K.: Ecasound (accessed June 2006). http://eca.cx/ecasound/.