根山 亮, 後藤 真孝, 村岡 洋一: ”WWW 上で公開可能な分散音楽情報処理システム— ダウンロードした Java アプレットのための安全なローカル通信方法 —”, 日本ソフトウェア科学会 第 14 回大会, B4-1, September 1997.

WWW 上で公開可能な分散音楽情報処理システム

— ダウンロードした Java アプレットのための安全なローカル通信方法 —

A Distributed Musical Information Processing System on WWW

— A method of secure local-communication for downloaded Java applets —

根山 亮　　後藤 真孝　　村岡 洋一
Ryo NEYAMA　　Masataka GOTO　　Yoichi MURAOKA
{ryo, goto, muraoka}@muraoka.info.waseda.ac.jp

早稲田大学 理工学部
School of Science and Engineering, Waseda University

概 要

分散音楽情報処理システムにおける各分散モジュールは, 演奏情報をリアルタイムに通信する必要がある. しかし, 現在の WWW と Java が提供する仕組みでは, WWW からダウンロードしたモジュール (Java アプレット) にはセキュリティ上の制約からローカル通信が許されておらず, すべての通信はダウンロード元のサーバを中継するため遅延が大き過ぎる. 本稿では, ダウンロードしたモジュールが安全にローカル通信と MIDI の入出力を行なうための枠組を提案し, 分散音楽セッションシステムへ応用した結果を述べる.

1 はじめに

近年, インターネットと WWW の急速な普及にともない, WWW 上での様々なアプリケーションや技術が開発されてきた. 特に Java [1] は WWW ブラウザ (以下, ブラウザと呼ぶ) 上でプログラムを実行する枠組を実現した. これによりブラウザは, 静的なデータだけでなく, 動画像・音楽なども扱えるようになった. また最近では, 音楽を演奏するための手段として, 音響信号だけでなく MIDI[†1] も扱えるようになってきた.

ブラウザ上で MIDI を扱うための従来手法のうち, Plug-in MIDI モジュールはダウンロードした標準 MIDI ファイル (SMF: *Standard MIDI File*) を再生するもので, Java アプレット (以下, アプレットと呼ぶ) からは利用できなかった. アプレットから SMF を再生するクラスライブラリ (Java Media Player [2]) も提案されているが, 仕様上, MIDI の自由な入出力はできなかった. MIDI の入出力を可能にするために, Java の native method を利用する方法 (Java MIDI [3]) も公開されているが, これは Windows95 の MIDI API を Java から利用するためのインターフェースで, 特定の計算機環境に依存しており, ダウンロードしたアプレットからはセキュリティ上の制約で利用できなかった.

これらの従来手法では, ダウンロードしたアプレットが MIDI 機器からの入力を読み込んだり, プログラムが MIDI イベントを動的に生成しながら演奏を出力することはできなかった. これは WWW を使って様々な音楽アプリケーションの開発する際には不十分である. 例えば, 計算機と人間とが合奏する音楽セッションシステムなどで, ユーザの演奏情報を受けとったり, その演奏に応じて自分の演奏を変化させる計算機の演奏者を実現することはできなかった.

本研究では, これらの問題を解決するため, ダウンロードしたアプレットから MIDI の入出力を動的に行なうためのシステム “Jam” を開発した. Jam では, MIDI 入出力をタイムスタンプ付きのイベントとして行なう汎用 MIDI クラスライブラリも提供することで, 様々な音楽情報処理システムの構築を容易にする.

さらに分散処理を実現するために, Jam 上のアプレットは, 同一セグメント上の他の同様なアプレットと安全にローカル通信し, 音楽情報の受け渡し

†1 *Musical Instrument Digital Interface.* 音源, キーボードなどの電子楽器の制御のための統一規格.

ができる．これは，Jamが提供するローカルディスク上の信頼できる通信ライブラリをロードし，アプレットがこの通信ライブラリと共有メモリ (Javaの静的変数) を介して音楽情報を受け渡すことにより実現した．

以下，**2** で本システムの概要について述べ，**3** で本システムを実現する上での課題を挙げる．そして，**4** でその解決法を示しながら，本システムの構成を説明する．**5** で本システムを用いたアプリケーションを紹介し，その実験結果を述べる．最後に **6** でまとめと今後の課題を述べる．

2 本システムの概要

近年，計算機とジャズセッションを行なうシステム [4]-[8] などの音楽情報処理システムに関する研究がなされてきた．本研究は，このような音楽情報処理システムの各処理モジュールを WWW 上からダウンロードし，LAN (Local Area Network) 上に分散した複数の計算機で動作させるシステムの実現を目的とする．

Jamは Java で記述されており，Jam 上の各分散モジュールはアプレットとしてブラウザ上で動作する．本システムでは，音楽情報を扱うためのインターフェースとして，電子楽器を制御するための統一規格 MIDI を用い，MIDI メッセージをネットワーク上で共有するための下位レイヤーのプロトコルとして RMCP[†2] [9] [10] を用いる．

Jamにより，アプレットが MIDI の入出力をできるだけでなく，例えば図1 のように，WWW 上に計算機演奏者を公開し，WWW 上の複数の計算機演奏者を一つの LAN 上にダウンロードしてセッションさせることが可能となる．また，VRML[†3]などと併用すると，演奏情報に合わせて動作する CG (*Computer Graphics*) 演奏者を実現できる．

3 システム実現上の課題

本節では，まずブラウザ上で動作するアプレットが受けるセキュリティ上の制約について説明し，次に本システム実現上の主要な課題を述べる．

(1) セキュリティ上の制約　WWWからダウンロードしたアプレットは，信頼できるプログラムであるとは限らない．そのためブラウザ上で動作するアプレットは，表 1 に示す制約を受ける [11]．

WWWからダウンロードしたアプレットはダウンロード元の計算機 (WWWサーバ) との通信のみ

†2 MIDI と LAN を融合した分散協調システムにおける，UDP/IP 上のコネクションレス型の通信プロトコル．

†3 *Virtual Reality Modeling Language*. WWW 上で 3 次元モデルを記述するためのプログラミング言語．

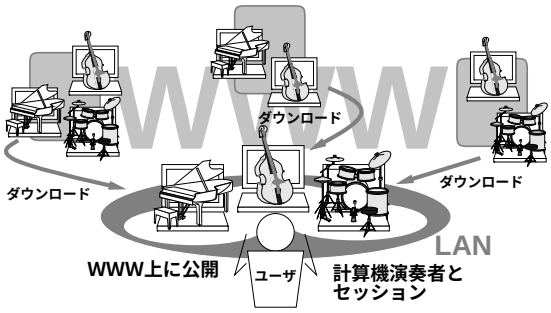

図1　WWW上の計算機演奏者

ダウンロード元	WWW	Local Disk
ファイルの読み書き	不可	不可
プロセスの起動	不可	不可
ライブラリのロード	不可	可
他の計算機との通信	制限付	無制限

表1　アプレットに対するセキュリティ上の制約

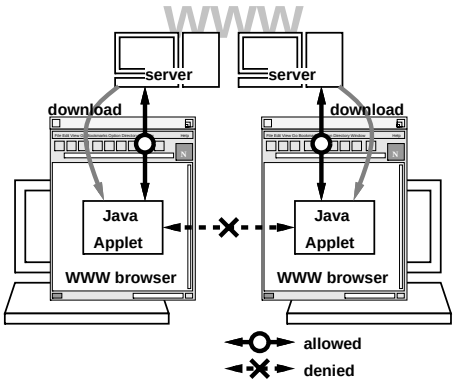

図2　WWW上のJavaアプレットの通信

許される．つまり図2 のように，WWW 上からダウンロードしたアプレット同士は直接通信できない．

(2) 本システム実現上の主要な課題　本システムの各分散モジュール (アプレット) は，音楽情報の入出力のための通信をリアルタイムに行なう必要がある．しかし，これらの通信をダウンロード元の計算機を経由して行なっていたのでは遅延が大き過ぎる．

そこで，Ethernet などの LAN 上でリアルタイムにローカル通信を行なう必要があるが，前述の通りWWWからダウンロードしたアプレットはダウンロード元の計算機以外との通信を許されていない．そのため，安全性を確保しつつ，分散モジュールが LAN 上でローカル通信を行なうための仕組みを考案する必要がある．

4 課題の解決法とシステムの構成

本節では，まずこの課題を解決するための指針を述べ，次に我々が実現したシステムの構成を説明する．

安全性を確保するには, 以下の 3 つの条件を満たすことが必要である.

- **通信のアドレス・ポート番号を限定する**— アドレス・ポート番号を限定することにより, 不法な通信を禁止する. アドレス・ポート番号をアプレット側に自由に設定させると, 故意に特定のサーバやデーモンプロセスに接続し, セキュリティ上の問題を引き起こす可能性がある.
- **通信のプロトコルを限定する** — アプレットから利用できる通信プロトコルを音楽演奏用の RMCP のみにすることにより, 機密情報およびシステムを保護する.
- **通信量を音楽情報の通信に適切な量に制限する** — 大量のパケット送信によるネットワークの停止を防ぐ. Ethernet の場合, 大量の UDP パケットを送信すると, collision により通信不能になる.

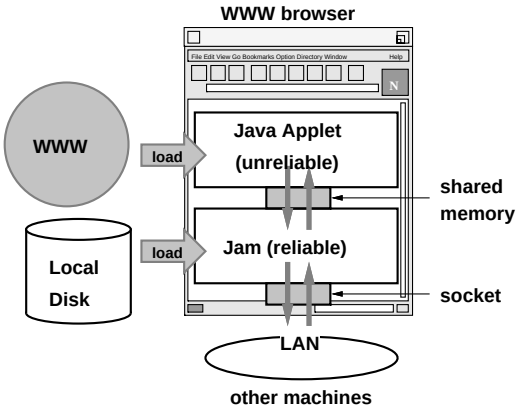

図 3　安全性確保の原理

我々は, 図3 のように, 音楽情報の送受信の役割を果たす Jam をアプレットと分離することにより, これらの条件を満たす枠組を提案する.

アプレットと Jam は, 異なるスレッドとして非同期に動作し, 共有メモリを介して音楽情報を受渡しする. Jam はローカルディスク上の信頼できるプログラムとして起動し, 通信量を監視しつつ適切なアドレス・ポート番号に対し RMCP を用いて通信するので, 安全性を確保できる.

上記の枠組を用い実装した Jam のシステム構成を図4 に示す. 図のように, Jam は “RMCP サーバ/クライアント (S/C)”, “Jam”, “RMCP メッセージ”, “アプリケーション” の 4 つのレイヤーから構成される. アプリケーションから Jam を隠蔽することにより, アプリケーション開発者は, Jam を意識することなく, RMCP のスタンドアローン・アプリケーションを開発する場合 (この場合, Jam レイヤーは不要) と同様に開発できる.

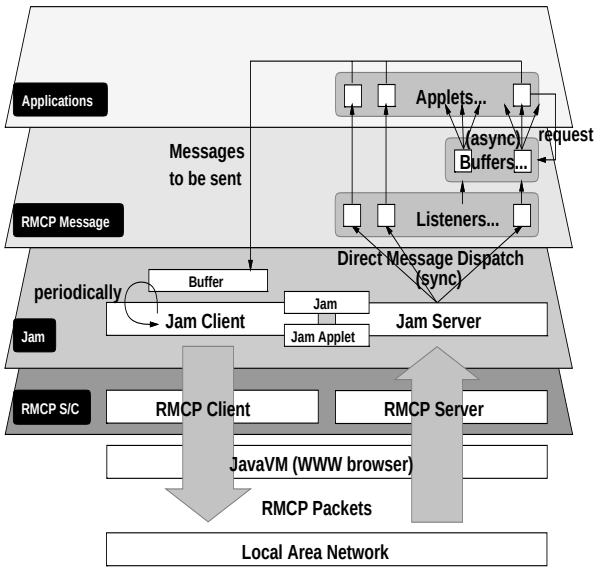

図 4　Jam のシステム構成

“RMCP サーバ/クライアント” レイヤーは, RMCP による通信を行なう. RMCP パケット (RMCP で受け渡される通信パケット) は, MIDI メッセージやタイムスタンプなどの情報を含む. RMCP クライアントが送信した RMCP パケットを RMCP サーバが受信することにより, それぞれの分散モジュール間で音楽情報を共有できる. MIDI を入出力するためには, 事前に入力用 RMCP クライアント (ユーザの演奏情報である MIDI 入力 を RMCP パケットとして送信) と出力用 RMCP サーバ (RMCP パケット中の MIDI メッセージをタイムスタンプの時刻に MIDI 機器へ送信) を起動しておく.

“Jam” レイヤーは, “RMCP サーバ/クライアント” レイヤーと “RMCP メッセージ” レイヤーの間に介在することにより, RMCP メッセージの受渡しを行なうと同時に, 通信量を監視・制限する. また, RMCP サーバ/クライアントの送受信する RMCP メッセージのみを扱い, 不法な通信を防ぐ.

Jam はローカルディスク上から起動され, Jam サーバ/クライアントの 2 つのスレッドを起動する. Jam サーバは, RMCP サーバが受けとった RMCP メッセージを登録されているリスナー (後述) に配送する. Jam クライアントは, バッファに溜められた RMCP メッセージ (後述) を定期的に RMCP クライアントに渡し, 送信させる.

“RMCP メッセージ” レイヤーは, Jam から配送される RMCP メッセージを, 1) アプレットに直接, あるいは 2) バッファに溜めてアプレットの要求に応じて, 受け渡す. これらの方法はアプレット側か

ら選択できる．前者の場合, Jamにアプレット自身をリスナーとして登録し, 後者の場合, バッファをリスナーとして登録する．Jamは登録されたアプレットやバッファに対し, 自動的に RMCP メッセージを配送する．

“アプリケーション” レイヤーで Jamを利用するアプレットは, 複数であってもよく, アプレットの代わりに VRML のスクリプトノード (Java などのプログラムから動作を制御するためのインターフェース) であってもよい．これには1 つの通信ポートを同一ブラウザ上の複数のアプリケーションで共有できるという利点がある．アプレットは, RMCP メッセージを Jamのバッファに登録することにより, 送信できる．これは直接 RMCP クライアントが RMCP メッセージを送信する場合と同じインターフェースで実現されるので, アプリケーション開発者は Jamを意識する必要がない．

5 アプリケーションと実験結果

我々は, Jamを用いたアプリケーションとして, 分散音楽セッションシステム “VirJa Sesssion on Java” [12] (VirJa Sessionは仮想ジャズセッションシステムを意味する) を開発した．このシステムにより, ユーザが新しい計算機演奏者を容易に開発・カスタマイズしてWWW 上に公開したり, 計算機演奏者をWWW 上から自宅のパソコンのような身近な場所にダウンロードし, セッションを行なうことが初めて可能になった．

我々はVirJa Session上の計算機演奏者の実装例として, ジャズのピアノトリオのベーシストとドラマーを実装した．ユーザはピアノを演奏することにより, 計算機演奏者と即興演奏を楽しむことができる．Jamにより, このようなリアルタイムに MIDI を入出力する音楽アプリケーションが容易に開発できることを確認した．

実験ではWWWサーバからベーシストとドラマーの演奏理解・生成処理モジュール (計算機演奏者) をダウンロードし, それぞれ LAN 上の異なる計算機の Netscape Navigator 3.01 上で実行した．その結果, それぞれの計算機演奏者が, 人間のピアノの演奏情報と, もう一方の計算機演奏者の演奏情報を受信し, リアルタイムに自分の演奏を生成し, LAN 上に送信できた．また, Jamが異なる複数の計算機環境 (Windows95, Solaris 2.5, IRIX 5.3 など) で動作することも確認した．

6 おわりに

本稿では, WWWから各処理モジュールをダウンロードして実行できる分散音楽情報処理システム Jamについて述べた．Jamでは, アプレットから MIDI を入出力するだけでなく, WWWから LAN 上の複数の計算機上にダウンロードした分散モジュール同士が, 音楽情報をローカル通信により共有し, 効率的に処理することを可能にした．またJamを利用した分散音楽セッションシステムを構築することにより, Jamがこのような音楽アプリケーションに有効であることを実証した．さらに, 一般の音楽アプリケーションは MIDI を入出力としているものがほとんどなので, Jamは汎用的に利用可能であると言える．

今後は Jamの一般公開に向けて, ドキュメント, ソースコードの整備を行なう予定である．

参考文献

[1] *Java:* `http://java.sun.com/`

[2] *Java Media Framework:* `http://java.sun.com/products/java-media/jmf/`

[3] *Java MIDI:* `http://www.users.interport.net/~mash/javamidi.html`

[4] 和気 早苗, 加藤 博一, 才脇 直樹, 井口 征士: **テンション・パラメータを用いた協調型自動伴奏システム**: *JASPER*, 情報処理学会論文誌, Vol.35, No.7, pp.1469-1481 (1994).

[5] 金森 務, 片寄 晴弘, 新美 康永, 平井 宏, 井口 征士: **ジャズセッションシステムのための音楽認識処理の一実現法** 情報処理学会論文誌, Vol.36, No.1, pp.139-152 (1995).

[6] 渡辺 和之, 西嶋 正子, 柿本 正憲, 村上 公一: **ニューラルネットワークを用いたジャズセッションシステム** — **ニューロミュージシャン** —, 第 44 回情報処理学会全国大会, 4R-5 (1992).

[7] 後藤 真孝, 日高 伊佐夫, 松本 英明, 黒田 洋介, 村岡洋一: **すべてのプレーヤーが対等なジャズセッションシステム** *I.* **システムの全体構想と分散環境での実装**, 情報処理学会研究報告, Vol.96, 音楽情報科学 96-MUS-14-4 (1996).

[8] 日高 伊佐夫, 後藤 真孝, 村岡 洋一: **すべてのプレーヤーが対等なジャズセッションシステム** *II.* **ベーシストとドラマーの実現**, 情報処理学会研究報告, Vol.96, 音楽情報科学 MUS-14-5 (1996).

[9] 後藤 真孝, 橋本 裕司: *MIDI* **制御のための分散協調システム** — **遠隔地間の合奏を目指して** —, 情報処理学会研究報告, Vol.93, 音楽情報科学 93-MUS-4-1 (1993).

[10] 後藤 真孝, 根山 亮, 菊地 淑晃, 村岡 洋一: *RMCP: Remote Media Control Protocol* — **時間管理機能の拡張と遅延を考慮した遠隔地間の合奏** —, 情報処理学会研究報告, Vol.97, 音楽情報科学 97-MUS-21-3 (1997).

[11] *Frequently Asked Questions — Applet Security:* `http://java.sun.com/sfaq/`

[12] 根山 亮, 後藤 真孝, 村岡 洋一: *WWW* **上の異なる計算機環境で動作する音楽セッションシステム** — VirJa Session on WWW **へ向けて** —, 第 54 回情報処理学会全国大会, 7J-5 (1997).