研究ノート

Basic Graphicsを用いた 初学者向けネットワークプログラミング

マッキン ケネス ジェームス*

要旨：ネットワークプログラミングを初学者に教える時，ネットワークプログラムの仕組みの基礎となるソケット通信から始めるのが望ましい．しかし，ソケットプログラミングを教える場合，サーバ側およびクライアント側の通信ソケットの生成処理の説明に加え，サーバ側でのクライアント接続待ちループや別スレッド生成によるクライアント接続確立など，最初のネットワークプログラムを完成するまでの工程が多く，ネットワークプログラムの初学者には難易度が高くなる．
本論文では，ネットワークプログラミングの初学者に対して通信手順（プロトコル）設計・実装の学習をスムーズに行うため，簡易的ソケット通信機能の提案を行う．具体的には，ソケット通信の仕組みを残しつつ，ソケット通信の確立までを極力簡易化し，かつ通信データを整数（int）型のみに制限した簡易ソケット通信機能を提案する．これにより，ソケット通信で重要となる，アプリケーション層での通信プロトコルの設計および実装にすぐに取り掛かることができ，初学者のネットワークプログラミングの理解と学習意欲の向上が期待できる．

キーワード：Java言語，ソケット通信，アプリケーション層，通信プロトコル，ライブラリ

Network programming for beginner learners using Basic Graphics

Kenneth James MACKIN*

Abstract: When teaching network programming, it is ideal to start with the basics using socket programming, yet socket programming requires various steps in creating the socket connection between the server and client, as well as understanding threading within the server and various error handling. This can be a drawback for beginners when starting socket programming. This paper introduces the network features of Basic Graphics, a Java programming library designed for beginner programmers, aimed at simplifying socket programming. Basic Graphics purposely only supports integer data types for data transfers, so that users can focus on designing the network protocols between client and server. By using Basic Graphics, learners are relieved from the technical details of setting up and maintaining socket connections, enabling the users to quickly get to the heart of network programming, to improve both understanding and learning motivation of the learners.

Keywords: Java language, Network socket, Application layer, Wire protocol, Programming library

* 東京情報大学　総合情報学部
Faculty of Informatics, Tokyo University of Information Sciences

2022年５月13日受付
2022年12月12日受理

１．はじめに

ネットワークプログラミングを初学者に教える時，ネットワークプログラムの仕組みの基礎となるソケット通信から始めるのが望ましい．しかし，ソケットプログラミングを教える場合，サーバ側およびクライアント側の通信ソケットの生成処理の説明に加え，サーバ側でのクライアント接続待ちループや別スレッド生成によるクライアント接続確立など，最初のネットワークプログラムを完成するまでの工程が多い. そのため, ソケット通信を用いたネットワークプログラミングで重要となる，アプリケーション層の通信プロトコルの設計および実装方法の説明に十分な時間が取れなくなる恐れがある．

本論文では，ネットワークプログラミングの初学者に対して通信手順（プロトコル）設計・実装の学習をスムーズに行うため，簡易的ソケット通信機能の提案を行う．

具体的には，ソケット通信の仕組みを残しつつ，ソケット通信の確立までを極力簡易化し，かつ通信データを整数（int）型のみに制限した簡易ソケット通信機能を提案する．簡易的ソケット通信機能をネットワークプログラミング教育に利用することで，初学者でも簡単にソケットプログラミングを行える．これにより，ソケット通信で重要となる，アプリケーション層での通信プロトコルの設計および実装にすぐに取り掛かることができ，初学者のネットワークプログラミングの理解と学習意欲の向上が期待できる．提案する簡易的ソケット通信機能は，初学者向けJavaプログラミングライブラリBasic Graphics[11][12]を拡張して組み込む．

２．TCP/IPプロトコルスタックとソケット通信

現在のインターネットの元祖となる，アメリカ国防総省のARPANET（Advanced Research Projects Agency NETwork）は，世界初のパケット通信を用いたWAN（Wide-Area Network：広域ネットワーク）であり，1969年の稼働当初のホスト間通信プロトコルは, 1822プロトコルと呼ばれた．1822プロトコルは，ホストアドレス，メッセージタイプ，データフィールドのみが規定されていた．

1978年には，インターネットワーキング用の通信プロトコルとして，上位層となるTransmission Control Program（TCP）と下位層となるInternet Protocol（IP）の二つ層に明確に分かれたTCP/IPプロトコルが作成された．TCP/IP（IPv4）は1983年にはARPANETの正式なプロトコルとなった．

TCP/IPは４層からなるプロトコルスタックを持つ．最下層の第１層のリンク層（link layer）は, イーサネットやWi-Fiを含む．第２層のインターネット層（internet layer）はIP（IPv4，IPv6）が担う．第３層のトランスポート層（transport layer）にはTCPやUDPがある．最上位の第４層のアプリケーション層（application layer）は，FTP，SSH，SMTPやHTTPなどのアプリケーション毎に定義される論理的な通信層である．

同じく1983年には，UC Berkeley（カリフォルニア大学バークレー校）が開発した4.2 BSD Unix Operating Systemに初めてソケット（BSD socket）が実装された．以降，BSDソケットは，プロセス間通信用APIとして，インターネットアプリケーションの標準インタフェースとなった．

ソケットは，アプリケーションとポート番号をつなげる仕組みを提供する．ポート番号は，あるホストに届いた通信を，そのホストで動作する様々なネットワークサービスに正しく割り振るための番号であり，それぞれのネットワークアプリケーション（あるいはネットワークサービス）は，事前に決められた特定のポート番号の通信を受け取る．ソケットは，ホスト名，ポート番号を指定し，データ通信用のインタフェースをアプリケーションに提供する．つまり，ネットワークプログラムのアプリケーション層の通信プロトコルを実装するということは，一般的にはソケット通信を用いてアプリケーション同士がデータの送受信を行うプログラムを作成することに他ならない．

３．ネットワークプログラミング教育

情報通信技術が社会インフラとして重要な位置を占め，情報化社会がさらに進む中，ICT教育の重要性がさらに増している．このような背景において，文部科学省も教育指導要領の見直しにおいて，初等中等教育におけるコンピュータネットワークの学習の強化を進めている．しかし，実際に初等中等教育の現場において授業の中でネットワークプログラミ

ングに十分な時間を取ることは難しいことが予想される．

文部科学省の高等学校情報科「情報Ⅱ」教員研修用教材（本編），第４章「情報システムとプログラミング」［１］では，ネットワークプログラミングに関して次のような記述がある．

- *既存のモジュールを再利用し，外部のリソースを効果的に活用した制作を行う．*
- *プログラムを制作しやすくするため，WebAPIなどの組み込み関数やあらかじめ用意した関数を利用できるようにする．*
- *開発は最低限の情報システムの制作にとどめ，実際に使用した上で，サイクルをかけながら機能を拡張していくことを想定して行う．*

このように，高校の専門選択科目においても，概念理解を中心の理解を目的とし，複雑な通信処理を省くことを勧めている．

大学等の高等教育機関でもネットワークプログラミングの需要が高い．中野ら［２］は，大学でのリモートセンシング講座の中での，Java言語を用いてネットワークで接続された機器同士の通信を行うネットワークプログラミングの実習結果について報告している．報告では，学生らはリモートサーバへの呼び出し方法，およびサーバ側単体での処理については理解できたが，クライアント-サーバ間の通信プログラムの実装に苦労し，通信ロジックを含めたネットワークプログラミングの学習方法に課題が残ったと報告している．

前述のような問題に対して，本研究では，ネットワークプログラミングの初学者を対象に，ネットワーク通信のトランスポート層以下の仕組みを極力意識せず，アプリケーション層の通信プロトコルの理解を主眼に置いた，Java言語用プログラミングライブラリの提案を行う．また，現在ネットワークシステム開発で広く利用されているJava言語のライブラリとして実現しているため，その後のより高度なネットワークプログラミング技術の学習にもスムーズに移行できる．

現在のネットワークシステムの開発は，Webアプリケーション・Webシステムが主流であり，かつ様々なWeb開発フレームワークが広く活用されている．また，それらサービスがクラウドプラットフォームを利用する場合も多い．これら高度に抽象化されたフレームワークを正しく理解し効果的に活用するためにも，ネットワークプログラミングの基礎をしっかりと理解することが重要となる．これは，用途別のライブラリを用いて抽象化されたプログラミングを行う場合においても，その下地となる言語の仕組みやアルゴリズムの特徴を理解することで，より効果的なソフトウェア開発が実現できるのと同じである．そのため，本研究では，ネットワークプログラミングにおけるアプリケーション層通信プロトコルの仕組みの理解に主眼を置く．

４．既存の言語・通信ライブラリとの比較

ネットワークプログラミング教育は，多くのプログラミング言語やライブラリを用いた報告がある．ここではいくつかを取り上げ，本研究での提案手法との違いとその意義を説明する．

Java言語は，今なおシステム開発で根強く支持されているプログラミング言語であり，特にサーバサイドの開発に幅広く利用されている．Java言語は，発表当初から言語の仕様としてネットワーク通信の機能を組み込んであることが特徴であり，そのため多くのネットワークシステムの開発に利用されてきた．Javaの標準機能としてのネットワーク機能は，大きく分けると，次の４種類が挙げられる．

１．Socket/ServerSocketによるTCP通信機能
２．DatagramSocketによるUDP通信機能
３．URL/URLConnectionによる通信機能
（HttpURLConnectionによるHTTP通信も含む）
４．RMI（Remote Method Invocation）によるリモートプロシージャコールによる通信機能

しかし，Javaはオブジェクト指向言語であるため，これら通信機能の実装には，オブジェクト指向の理解に加え，ソケットクラス，入出力ストリームクラス，例外処理，スレッド処理などの理解が必要となるため，実際のアプリケーション層通信プロトコルの検討に入る前の処理が複雑で多い．これらJava標準のソケットクラスの利用に比べると，本提案にあるプログラミングライブラリの簡易的通信機能を用いることで，すぐにネットワークプログラムのアプリケーション層通信プロトコルの実装に入ることができ，ネットワークプログラミングの概要や仕組みが理解しやすくなる．

Processing［３］［４］は，Javaをベースとして使いや

すさを目指したプログラミング言語であり，本研究の提案とほぼ同じ観点から，簡易的な通信ライブラリ（processing.net.*）でソケット通信を標準でサポートする．Processingを用いる場合のデメリットとしては，標準入力の機能が無いため，文字列の入力を行うコンソールプログラム作成することができない．また，Javaに文法が似ているが，様々なところで独自のプログラミング作法を用いているため，本格的なネットワークプログラム開発を行う時には，別の言語を学びなおす必要がある．

ドリトル[５]は，Javaベースの日本語テキストプログラミング言語であり，ドリトルの分散オブジェクト機能[６]を用いたネットワークプログラミング教育が報告[７]されている．しかし，ドリトルの分散オブジェクト機能は，Java RMIを用いた通信処理であり，通常のTCPまたはUDP通信で用いられるソケット通信機能とは違うため，アプリケーション層通信プロトコルの学習に主眼を置く本研究の目的とは合致しない．

Microsoft MakeCode[８][９]は，教育でも広く利用されているプログラミング学習環境であり，その特徴として，様々なマイクロコンピュータでの実行が可能であることが挙げられる．コード作成は，ブロックを用いたブロックプログラミングや，ブロックと同じ命令を用意したJavascriptコーディングが選べる．MakeCodeでは無線通信を標準でサポートしているため，通信機能を備えたMakeCode対応マイクロコンピュータ同士でMakeCodeの無線通信を実行すると，２台のマイクロコンピュータ間でデータ通信を実現できる．しかし，MakeCodeの無線通信は，通信の受信をイベントコールバックとしてのみ実現しているため，通信処理の途中で通信相手からの受信を待って同期を取ることが難しい．

Pythonも教育で広く利用されているプログラミング言語であり，標準でJavaと同等のソケット通信の機能を持っている．また，教育向けにPythonのソケット通信処理を簡易的に利用できるラッパーモジュールが報告[10]されている．Pythonも近年ネットワークシステム開発に利用され始めているため，ネットワークプログラミング教育にPythonを利用するメリットはある．ネットワーク機能の使いやすさにおいては，本提案のJavaを用いた簡易的なソケット通信と同等である．また，PythonにもTcl/Tk GUIツールキットへのインタフェース用ライブラリtkinterなどが用意され，Java言語と同等のGUI機能がある．ただし，本提案で拡張したJava用ライブラリBasic Graphicsは，Java言語標準のGUI機能よりも初学者に使いやすい簡易的なグラフィックス機能が含まれるため，通信処理のための入出力のおよび通信結果の描画の容易さにおいて本提案に優位性がある．これにより，GUIプログラミング教育が進んでいない学生に対しても，ネットワークプログラミングを用いたグラフィカルアプリケーションの作成も可能である．

また，Javaは現在でもシステム開発の主流言語のひとつであり，そのためJava言語をプログラミング教育に用いているカリキュラムも多い．本提案の簡易ソケット通信機能はJava用ライブラリとして実装しているため，プログラミング教育をJavaで進めている教育環境にとっては特に親和性が高い．

５．Basic Graphics ネットワーク機能

本研究では，Java言語教育向けの簡易グラフィックス機能を中心とした，プログラミング初学者向けのJavaクラスライブラリBasic Graphics[11][12]を拡張し，簡易ソケット通信機能を追加した．Basic Graphicsの基本設計思想は，あえて機能性を絞り込むことで，シンプルで分かりやすく，使いやすいライブラリを目指している．また，Basic Graphicsは，プログラミング学習の初期段階でのコンソールで実行されるテキストベースプログラミングの時点からスムーズに利用できるよう，テキストベースプログラミングとの親和性を考慮している．

Basic Graphicsの特徴として，オブジェクト指向を極力隠蔽することで，オブジェクト指向の学習がまだの初学者や，オブジェクト指向が必要ない短い実験プログラムやプロトタイプ作成が目的の利用者が，簡単に扱えることができる．特に，グラフィックス表示を行うために通常必要となるウィンドウオブジェクト操作，非同期イベントハンドリング，描画タイミングやスレッド処理などをまったく意識せずに利用できる．

簡易グラフィックス機能の他にも，リアルタイムキー入力，マウス入力，サウンド再生，音符演奏，ファイル入出力等が備わっている．

6．Basic Graphicsネットワークプログラム

6．1 ソケット通信プログラム

本論文での提案となる，Basic Graphicsを拡張したネットワークプログラミングについて説明する．Basic Graphicsを用いたネットワークプログラミング教育は，トランスポート層以下の処理を隠蔽し，ユーザにアプリケーション層通信プロトコルの設計および実装に集中させることが目的となる．

Basic Graphicsクラスに追加した主要な通信用メソッドを以下に説明する．

public boolean connect（String hostname, int port）

指定したホスト・ポート番号にソケット接続を行う．サーバ側（接続待ち側）を生成するには，第一引数のhostnameをnullにし，ポート番号のみ指定する．クライアント側を生成するには，ホスト名とポート番号を指定する．ホスト名には，名前解決可能なホスト名またはIPアドレスの文字列が利用できる．具体的なホスト名の例として，“localhost”，“127.0.0.1”や，“aelab.tuis.ac.jp”などのFQDN（Fully Qualified Domain Name）文字列が利用できる．メソッドは接続が確立されるまで待ち状態になり，接続に成功すると true を返し処理を続行する．接続でエラーが起きた場合は，falseが返る．

public boolean send（int message）

接続されたソケットに対して，引数の整数値を送信する．ソケットに対してデータの送信を完了すると，通信相手の読み出しを待たず，ただちにtrueを返し処理を続行する．送信でエラーが起きた場合は，falseが返る．

public int receiveint（）

接続されたソケットに対して，整数データを受信し，読み込んだ整数値を返す．ソケットに対して受信データが無い場合，データが届くまで待ち状態となり，処理を終了しない．

public int scanint（）{

接続されたソケットに対して，整数データを受信し，読み込んだ整数値を返す．ソケットに対して受信データが無い場合，receiveint（）とは違い待ち状態にならず，ただちにEOF（End Of File）を返して処理を続行する．

上記以外にも，サーバに接続した複数のクライアントをインデックス番号で指定して，個別に通信を行うためのメソッドもあるが，ここでは説明を省く．

拡張したBasic Graphicsクラスを用いた，クライアント－サーバ間のデータ送受信の例を示し，利用方法を説明する．

サーバ側のプログラムをリスト１に示す．

リスト１　サーバ側プログラム

```
import jp.ac.tuis.basic.*;
public class Server {
  public static void main(String args[]){
    BasicGraphics g = new BasicGraphics();
    int port = 50001;
    g.println("Waiting for connection to
port:"+port);
    g.connect(null,port);       //クライアント接続待ち

    g.println("Connected!");
    int data = g.receiveint(); //データを受信
    g.println("Received:"+data);
    data=data+10;
    g.println("Sent:"+data);
    g.send(data);               //データを送信
    g.println("End");
  }
}
```

クライアント側のプログラムをリスト２に示す．

リスト２　クライアント側プログラム

```
import jp.ac.tuis.basic.*;
public class Client {
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    String host = "localhost";
    int port = 50001;
    g.println("Connecting to "+host+":"+port);
    g.connect(host, port);   //サーバへ接続

    g.println("Connected!");
    int data=10;
    g.println("Sent:"+data);
    g.send(10);              //データを送信
    data = g.receiveint();   //データを受信
    g.println("Received: "+data);
    g.println("End");
  }
}
```

両プログラムのコンパイルは，Windowsの場合は次のように行う．

```
javac -cp .;basic.jar Server.java
javac -cp .;basic.jar Client.java
```

実行は，サーバを先に実行する．ここでは，コンソール（コマンドプロンプト等）から二つのJavaプ

ロセスを順に実行したいため，javaコマンドではなくjavawコマンドを利用する．二つのコンソール（コマンドプロンプト等）からそれぞれJavaプロセスを実行する場合，javaコマンドで問題ない．

```
javaw -cp .;basic.jar Server
javaw -cp .;basic.jar Client
```

サーバプログラムServerとクライアントプログラムClientの実行結果を示す（図１，２）．Clientが送信したデータ（10）がサーバ側で更新され，Clientが更新されたデータ（20）を受け取っていることが分かる．

前記Client-Serverプログラムの通信手順をシーケンス図（図３）で示す．

上記プログラム例では，まずServerでconnect（port）でClientからの接続待ちになる．ClientからServerに対してconnect（host, port）を呼ぶことで，Serverの待ち状態が終了し，ClientとServer共に動き出す．Client側が先にsend（int）を行い，Server側はreceiveint（）で送られた数値（10）を読み出す．次にServer側がsend（int）を行い，Client側がreceiveint（）で送られた数値（20）を読み出す．

シーケンス図でも見られるように，ネットワークで接続したプログラムが，どちら側が送り側でどちら側が受け取り側，そしてそれぞれ何を送るのか，という通信手順を決めるのがアプリケーション層通信プロトコルの役割である．この通信手順を間違うと，両方待ち状態になりプログラムが動かなくなる，期待と違う情報を受け取り処理が正しく行えない，情報が足りないことで処理を継続できない，などの問題が発生してしまう．

拡張したBasic Graphicsでは，あえて送受信でき

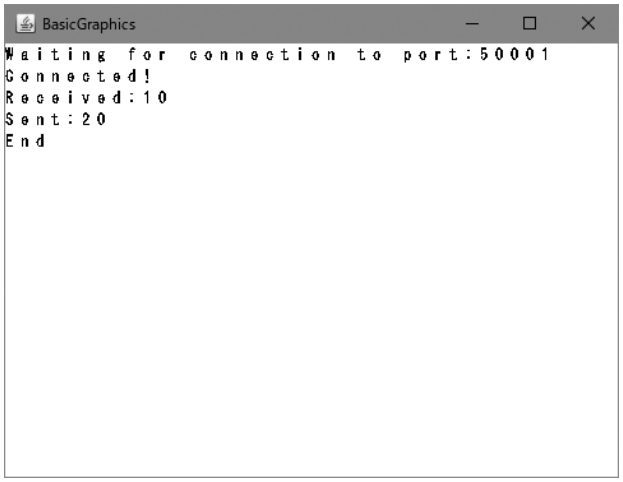

図１　Serverプログラムの実行結果
Figure 1　Result of the Server program.

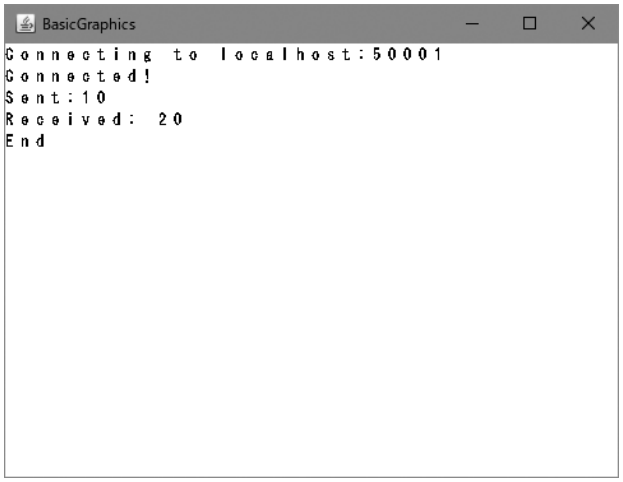

図２　Clientプログラムの実行結果
Figure 2　Result of the Client program.

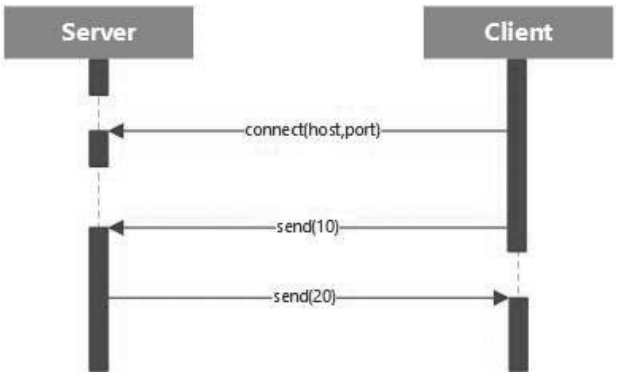

図３　Client-Serverプログラムシーケンス図
Figure 3　Sequence diagram for the Client-Server communication.

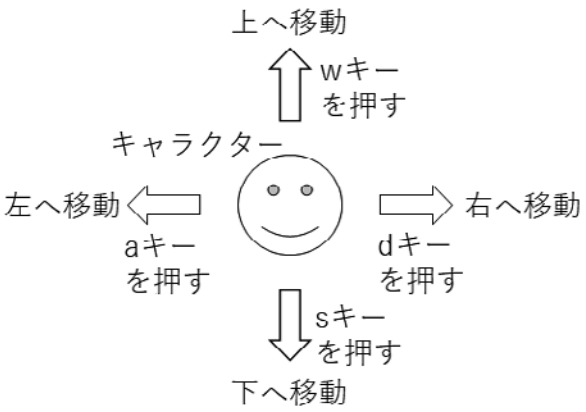

図４ キー操作とキャラクター移動の関係
Figure 4　Relationship of key press and character motion

るデータを整数型（int型）のみとすることで，送受信するデータのタイミング，回数，およびデータ内容に注意を払い，アプリケーション層通信プロトコルの理解を促すことを目的とする．

Basic Graphicsを用いることで，従来のコンソールベースのテキストプログラムに簡単にネットワーク通信機能を追加できる他，グラフィックスやリアルタイムキー入力，マウス入力なども利用できるため，様々なグラフィカルアプリケーションも簡単にネットワーク通信機能を追加することが可能となる．それにより，学習者に様々なパターンのアプリケーション層通信プロトコルを考える課題を用意でき，学習者の学習意欲を維持しつつ，理解度を高めることが可能となる．

6.2 リアルタイムアニメーションネットワークプログラム

次に，簡単なユーザ入力を用いたアニメーションプログラムに通信機能を追加する例を示す．プログラムは，アクションゲームのキャラクター操作のように，ユーザによる特定のキー（A，S，D，W）の押下タイミングに合わせて，画面上のキャラクターが，キーの示す方向に画面上をリアルタイムで移動（アニメーション）する（図４）．キー操作によるアニメーション例題は，学習者の興味を引き，学習意欲を高めるために用いる．

このプログラムをネットワークプログラムとして，片方をクライアントとして，もう片方をサーバとして別々のコンピュータで実行する．さらに，自分の画面には，自分の操作するキャラクターに加えて，相手の操作するキャラクターも表示する．つまり，それぞれの画面には，自分と相手の操作するキャラクターがリアルタイムでアニメーション動作する．今回はゲーム要素を加えていないが，これはリアルタイムアクションゲームのネットワーク対戦の仕組みを簡略化したモデルと考えることができる．

サーバ側のプログラムソースコードをリスト３に示す．

リスト３　サーバ側アニメーションプログラム

```
import jp.ac.tuis.basic.*;
public class AnimationServer{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    g.console(false);

    int x=30;
    int y=10;
    int remotex=10;
    int remotey=10;
    char key=0;
    char remotekey=0;
    long SLEEP=10;

    //ネットワーク接続
    int port = 50001;
    g.println("Waiting for connection to
port:"+port);
    g.connect(null, port);

    //画面用意
    g.cls();
    g.locate(x,y);
    g.color(g.BLUE);
    g.print("B");
    g.locate(remotex,remotey);
    g.color(g.CYAN);
    g.print("A");

    //メインループ
    while(true){
      g.sleep(SLEEP);//スピード調整

      //ユーザキー入力
      key = g.inkey();

      //同期を取る
      remotekey = (char)g.receiveint(); //データを受
取
      g.send(key); //データを送る

      //プレイヤー操作
      g.locate(x,y);
      g.print((char)0);
      if(key == 'a' && x > 0)  x--;
      if(key == 'd' && x < 39) x++;
      if(key == 'w' && y > 0)  y--;
      if(key == 's' && y < 19) y++;
      g.locate(x,y);
      g.color(g.BLUE);
      g.print("B");

      //対戦プレーヤーの操作
      g.locate(remotex,remotey);
      g.print((char)0);
      if(remotekey == 'a' && remotex > 0)  remotex-
-;
      if(remotekey == 'd' && remotex < 39)
remotex++;
      if(remotekey == 'w' && remotey > 0)  remotey-
-;
      if(remotekey == 's' && remotey < 19)
remotey++;
      g.locate(remotex,remotey);
      g.color(g.CYAN);
      g.print("A");
    }//end while(true)
  }//end main
}//end class AnimationServer
```

次にクライアントプログラムソースコードをリスト４に示す．ほぼ同じ処理だが，send（）とreceiveint（）の順番が違う．

リスト4　サーバ側アニメーションプログラム

```
import jp.ac.tuis.basic.*;
public class AnimationClient{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    g.console(false);

    //変数初期化
    int remotex=30;   //相手プレイヤーx 座標
    int remotey=10;   //相手プレイヤーy 座標
    int x=10;         //プレイヤーx 座標
    int y=10;         //プレイヤーy 座標
    char key=0;       //プレイヤーキー入力
    char remotekey=0; //相手プレイヤーキー入力
    long SLEEP=10;    //待ち時間(millisec)

    //ネットワーク接続
    String servername="localhost";
    int port=50001;
    g.println("Connecting to
"+servername+":"+port);
    g.connect(servername, port); //サーバに接続

    //画面用意
    g.cls();         //画面を消す
    g.locate(x,y);   //座標設定
    g.color(g.CYAN); //色設定
    g.print("A");    //文字表示
    g.locate(remotex,remotey); //座標設定
    g.color(g.BLUE);           //色設定
    g.print("B");              //文字表示

    //メインループ
    while(true){
      g.sleep(SLEEP);//スピード調整
      key = g.inkey();//ユーザキー入力

      //同期を取る
      g.send(key); //データを送る
      remotekey = (char)g.receiveint(); //データを受
取

      //対戦プレーヤー操作
      g.locate(remotex,remotey);
      g.print((char)0); //文字を消す
      if(remotekey == 'a' && remotex > 0)  remotex-
-;
      if(remotekey == 'd' && remotex < 39)
remotex++;
      if(remotekey == 'w' && remotey > 0)  remotey-
-;
      if(remotekey == 's' && remotey < 19)
remotey++;
      g.locate(remotex,remotey);
      g.color(g.BLUE);
      g.print("B");

      //プレイヤー操作
      g.locate(x,y);
      g.print((char)0); //文字を消す
      if(key == 'a' && x > 0)  x--;
      if(key == 'd' && x < 39) x++;
      if(key == 'w' && y > 0)  y--;
      if(key == 's' && y < 19) y++;
      g.locate(x,y);
      g.color(g.CYAN);
      g.print("A");
    }//end while(true)
  }//end main
}//end class AnimationClient
```

今回のネットワークプログラムのシーケンス図（図５）を示す.

リアルタイムアニメーションネットワークプログラムの実行画面（図６）を示す．AnimationServerとAnimationClientの画面表示はまったく同じため，AnimationServerのみ示す.

このネットワークプログラムのアプリケーション層通信プロトコルは，自分が動いた方向を相手側に伝えることで，お互いの位置情報を同期させ，同じ画面を共有することが出来るようになっている．また，お互いが交互にreceiveint（）で相手の通信処理を待つことで，リアルタイムの同期を保証している.

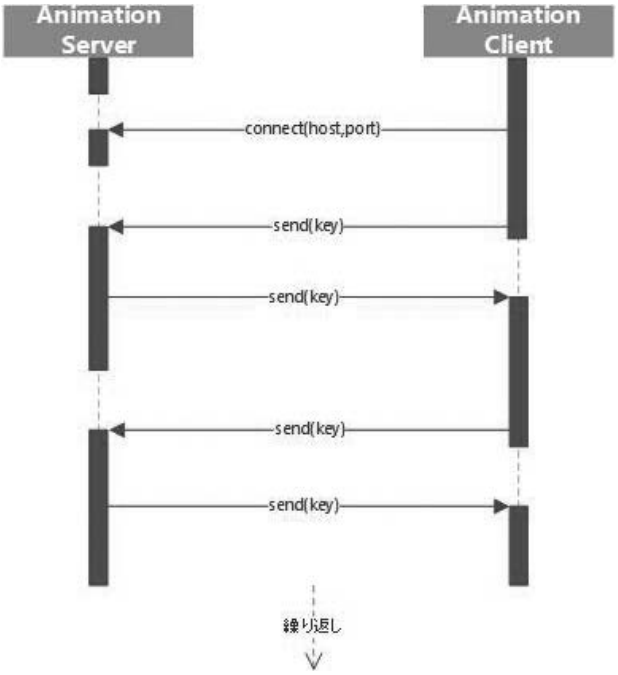

図５　アニメーションプログラムシーケンス図
Figure 5　Sequence diagram for the network animation program.

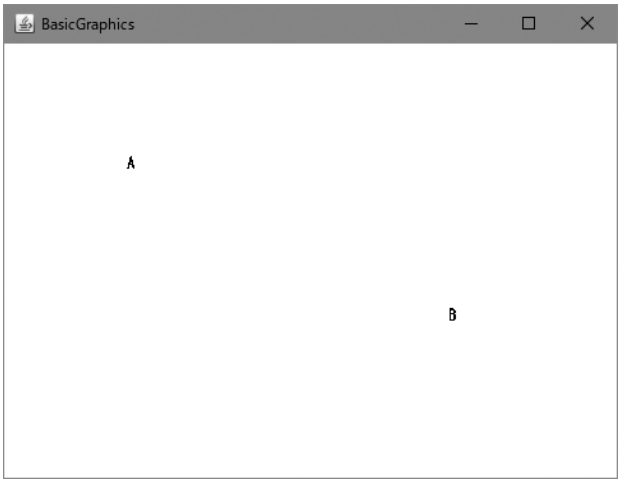

図６　AnimationServerプログラムの実行結果
Figure 6　Result of the ServerServer program.

表1 アンケート集計結果
Table 1 Student survey results

	賛 成		やや賛成		どちらとも言えない		やや反対		反 対		合計	
質問1	8	35%	13	57%	1	4%	0	0%	1	4%	23	100%
質問2	7	30%	13	57%	2	9%	0	0%	1	4%	23	100%
質問3	11	48%	10	43%	1	4%	0	0%	1	4%	23	100%
質問4	10	43%	11	48%	0	0%	0	0%	2	9%	23	100%

6.3 アプリケーション層通信プロトコル設計

アプリケーション層通信プロトコルの設計において，通信のシーケンス制御，つまりどちらが受信してどちらが送信するか，と共に重要となるのが，通信データの設計となる．本提案のBasic Graphics拡張では，あえてint型の通信のみに制限することで，構造化データのシリアル化処理なども自ら実装する必要があり，そのことでアプリケーション層通信プロトコルの理解度を高めることも念頭においている．

通信データ設計の具体例として，アプリケーション層の構造体データの通常の通信設計では，最初のデータは通信種別（あるいはデータ種），次にデータサイズ，その後にデータが続く．

これを拡張したBasic Graphicsで実装する場合，最初にデータ種をint型で送信し，次にデータサイズ（int型データの数）をint型で送信し，その後データサイズで指定した個数のint型データを送信する．文字などを送る場合は，一文字ずつchar型をint型に変換して送信し，受信側でchar型に戻す処理を行う．

拡張したBasic Graphicsでの実装例をリスト5，6に示す．

リスト5 文字列送信側プログラム

```
String s = "Hello";
send(1);                //1 は文字列データの送信
send(s.length());       //文字数を送る
for(int i=0; i<s.length(); i++){
    send(s.charAt(i)); //一文字づつ int 型で送る
}
```

リスト6 文字列受信側プログラム

```
int data = receiveint();
if(data==1){                //1 は文字列データの受信
    int len = receiveint(); //文字数受信
    for(int i=0; i<len; i++){
        System.out.print((char)receiveint());
        //一文字ずつ char 型として受信
    }
}
```

上記では，文字列データを通信する例を示したが，送信側と受信側でどのようなデータがどの順番で送られるかが事前に決まっていれば，あらゆる構造体データの通信ができる．つまりこれが通信プロトコル設計における通信データ設計に他ならない．

通信プロトコル設計においては，上記の通信データ設計と合わせて，通信の順番を決定する通信シーケンスを設計する必要がある．本実験では，クライアント-サーバ型通信の基本となる，1）クライアントからの問い合わせ，2）サーバからの回答，のトランザクションペアを基本的な通信パターンとした．

7．実験講義

実験講義の具体的な目標を以下に設定した．

1．ネットワークプログラミングの初学者が，1対1のクライアントサーバモデルにおいてのソケット通信を用いたアプリケーション層通信手順（プロトコル）の設計を理解できる．
2．ネットワークプログラミングへの学習意欲を高めることができる．

実験講義の教育設計を以下とした．

1．学習者のソケット通信の理解度確認
2．ソケット通信の解説およびソケット通信を用いたアプリケーション層通信プロトコルの考え方の講義
3．Basic Graphics簡易ソケット通信の利用方法の説明
4．Basic Graphics簡易ソケット通信の利用例（リアルタイムアニメーション）の説明
5．「じゃんけん」「あっちむいてホイ」「鬼ごっこ」等いくつか課題例を提案し，学生に自らソケット通信を用いたネットワークプログラムを作成
6．学生の理解度を確認するアンケートを行う

本提案で拡張したBasic Graphicsによるネットワークプログラミング教育の効果を確認するため，東京情報大学総合情報学部総合情報学科３年生13名４年生10名に対して実験講義を行った．学生は全員，１年次と２年次でJavaプログラミングの授業を受講しており，かつソケット通信を用いたネットワークプログラミングの経験は１人を除いて全員無かった．

最初に15分程度でソケット通信の仕組みとアプリケーション層通信プロトコルの考え方を講義し，その次15分程度で，本論文で示した最初のClient-Serverのプログラム例を用いて，拡張したBasic Graphicsのソケット通信の利用方法を講義した．講義の最後に，本論文で示したネットワークアニメーションプログラムの例を見せ，30分間学生に自由にネットワークプログラムを作成するように指示した．最後に下記に示すアンケートを行った．

アンケート項目

１．アプリケーション層通信プロトコルについて理解できたか
２．Basic Graphicsのネットワーク機能（send/receive）について理解できたか
３．今後Basic Graphicsネットワーク通信機能を使ってみたいか
４．Basic Graphicsの他の機能（音楽演奏機能，画像描画機能，立体表示機能など）に興味があるか
５．自由記述

１－４の項目は，それぞれ５段階評価（賛成，やや賛成，どちらとも言えない，やや反対，反対）として回答を求めた．

アンケートの結果を表１に示す．

理解度を確認する設問１，２では，賛成とやや賛成の合計が，全体の91％（問１）と87％（問２）を占め，参加した学生のほぼ全員がソケット通信の仕組みを理解できた．また，設問３，４も賛成とやや賛成の合計が全体の91％を占め，Basic Graphicsが学生の興味を引くことに成功した．これは，学生のネットワークプログラミングの学習意欲にポジティブな影響を及ぼしたと考えることができる．

なお，自由記述を見ると，問１，２共に全く理解できなかったと回答した学生が,「もっとネットワークについて勉強していきたいと思った.」と回答しており，短い実験講義で理解が不十分だった学生にも，学習意欲を高める効果があった．また，問３，４の両方において，Basic Graphicsにまったく興味がない，と回答した学生１名は，すでにネットワークプログラミングを独学で学習済みの学生であり，そのような学生にとっては簡易化したネットワーク機能が逆に不自由だったと考えられる．

今回の実験講義は，実際のネットワークプログラミングの講義に適用したものではなく，またその教育効果を試験などで定量的に計測していないため，提案した簡易ソケット通信機能を追加したBasic Graphicsの教育効果を明確に証明することはできていない．しかし，今回の実験講義の結果，目標１のアプリケーション層通信手順の理解については，アンケートにより91％の学生の理解が向上したことを確認できた．また，目標２のネットワークプログラミングの学習意欲の向上については，アンケートにより91％が簡易ソケット通信機能に興味を示し，また自由記述からもネットワークプログラミングへの興味が確認できた．よって提案手法は，ネットワークプログラミングに対して理解度を高め，さらに学生の興味を引くことには成功しており，学習意欲にはポジティブな影響を及ぼしたと考えることができる．

８．おわりに

本研究では，ネットワークプログラミングの初学者に向けた簡易的なソケット通信機能を提案し，Java用簡易グラフィックスライブラリBasic Graphicsに簡易ソケット通信機能を追加した．拡張したBasic Graphicsは，オブジェクト指向をほぼ隠蔽した形で，ソケット通信やグラフィックス機能をユーザに提供する．本論文では，拡張したBasic Graphicsの簡易的なソケット通信機能およびグラフィックス機能の利用方法を解説し，拡張したBasic Graphicsがアプリケーション層通信プロトコルの学習に利用できることを説明した．続いてJavaプログラミングを学んだ学生に対して行った実験講義の結果を報告した．実験講義のアンケート結果では，Basic Graphicsはネットワークプログラミングの理解度および学習意欲の向上に効果がある可能性を示すことができた．

また，本提案の拡張版Basic Graphicsは，大学などの高等教育機関だけではなく，高校などで新学習指導要領に準じてネットワークプログラミングを時間数の限られた授業で教える場合においても，実践的なネットワークプログラミングの学習に利用できるものと考えられる．

今後の開発課題として，拡張したBasic Graphicsを用いたネットワークプログラミングのオンライン教材がまだ少ないため，教材の充実を進める．

参考文献

［1］文部科学省：高等学校情報科「情報Ⅱ」教員研修用教材，<https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/mext_00742.html>（参照2022年５月13日）．

［2］中野裕司，永峰康一郎，横澤　肇，喜多敏博，秋山秀典：Telnetから始めるリモートセンシング実験，PCカンファレンス論文集（2004）．

［3］Reas, C. and Fry, B.: Processing: programming for the media arts, Ai & Society, Vol.20, No.4, pp.526-538（2006）．

［4］Processing.org, <https://processing.org/>（参照2022年５月13日）．

［5］大阪電気通信大学兼宗研究室：プログラミング言語「ドリトル」，<https://dolittle.eplang.jp/>（参照2022年５月15日）．

［6］兼宗　進，中谷多哉子，御手洗理英，福井眞吾，久野　靖：教育用プログラミング言語におけるオブジェクト共有機能の導入，情報処理学会論文誌プログラミング（PRO），Vol.45, No. SIG05（PRO21），p.81（2004）．

［7］西ヶ谷浩史：中学校におけるプログラミング教育―ネットワークを利用したプログラミングと計測・制御，情報処理，Vol.60，No.10，pp.1022-1028（2019）．

［8］Ball, T., Chatra, A., de Halleux, P., Hodges, S., Moskal, M., and Russell, J.: Microsoft MakeCode: embedded programming for education, in blocks and TypeScript, Proceedings of the 2019 ACM SIGPLAN Symposium on SPLASH-E（SPLASH-E 2019）, ACM, pp.7-12（2019）．

［9］Microsoft MakeCode, <https://www.microsoft.com/makecode>（参照2022年５月15日）．

［10］安本太一，大久保直樹，岡部直樹，磯部征尊：Pythonによる計測・制御とネットワークを利用した双方向性のあるコンテンツのプログラミング授業実践と評価，日本教育工学会研究報告集，No.1，pp.81-88（2021）．

［11］マッキン ケネス ジェームス：初学者教育用Javaプログラミングライブラリ「Basic Graphics」，東京情報大学研究論集，Vol.25，No.2，pp.35-44（2022）．

［12］マッキン ケネス ジェームス：プログラミング教育向けJavaライブラリBasic Graphics, <http://www.edu.tuis.ac.jp/~mackin/basic/>（参照2022年５月15日）．