５年工学実験テキスト(令和３年版)　　2021.05.19

宮武

ネットワークプログラミングの基礎と応用(VB2019)

目　次

目　的

- プロセス間通信，クライアント・サーバモデル，ソケットなど，ネットワークを利用するソフトウェア（以降では，ネットワークアプリケーションと呼ぶことにする）を開発するための基礎知識の修得
- ネットワークアプリケーションの設計・開発手順の修得
- Visual Basic 2019 によるイベント駆動型プログラムの作成およびデバッグなどに関する技術の修得

１．ネットワークプログラミング入門

1.1 ネットワークのレイヤモデル

ネットワーク接続されたコンピュータ間で，ファイルの交換を行う方法を提供するといった特定の作業が与えられたとき，通常この作業をより小さな部分に分割し，これらの分割された問題を解決する。分割された問題の解をあつめて，最終的な解を形成する。

ネットワーキングの世界では，このように部分的な問題への分割を行うことを**レイヤ化**(layering)と呼ぶ。通信の問題は，いくつかの部分（層－レイヤ）に分割され，各レイヤは特定の機能を提供することに専念する。

ネットワーク上のコンピュータは，厳密に定義された**プロトコル**（protocol）を用いて通信する。プロトコルとは，通信に参加するもの同士の規則や慣例の集まりである。

（1）OSI（Open Systems Interconnection）モデル

国際標準化機構（ＩＳＯ：International Standard Organization）で開発されたモデルで，図１に示すように７層からなる。

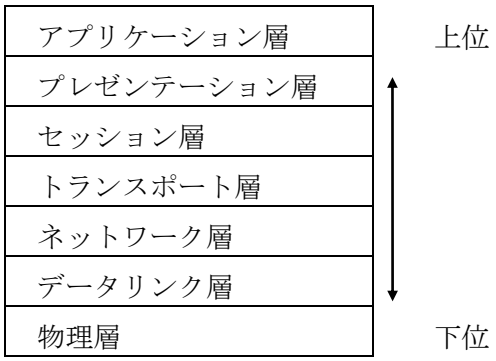


図１　ＯＳＩモデル

物理層

データを１と０の２進ビットで伝送するが，この１，０を実際にどのように表現するかに関する取り決めを行う層である。シリアル通信のための RS-232C という仕様がある。

「調査項目 1」RS-232C，ADSL，FTTH の機能，特徴について調べよ。

データリンク層

誤り検出，訂正機能や，データの論理的単位である**フレーム**に区分するための制御を行う層である。フレームとは，伝送情報に，送受信のタイミングをとるための情報，送信相手のアドレス情報，誤り検出情報などを付加したものである。

ネットワーク層

ネットワークの形態によって，自分の通信したい相手局が中継ノード経由で間接的につながっている場合がある。このとき，中継ノードは**パケット**が目的局に届くように中継しなければならない。ネットワーク層がこの操作を行う。ネットワーク層のフレームは，通

常パケットと呼ぶ。

a）データグラム（datagram: data+telegram による造語）

パケットごとに相手アドレスをつけ，バラバラに送出する方法。各パケットは，個別にネットワーク上を流れて目的局に届くため，送出時に順序が狂う可能性がある。

b）バーチャルコール（virtual call）

データ送出に先立ち，ネットワーク内に仮想的なデータ流路を設定する。次に，この通路上に連続的にデータパケットを流す。最後に，この流路を解放する。流路設定，解放の手間はかかるが，到着パケットを順序は必ず保証される。

トランスポート層

中継ノードの故障などにより，パケットが紛失してしまったり，同じパケットが重複して届いたりしないようにする。このように，ネットワークの誤動作をカバーするために導入された層である。

セッション層

通信を望んでいる遠隔地のプロセス間を直接結合するためのプロトコルサービスを行う。同一計算機内のプロセス間での通信は，プロセス間通信と呼び，従来からオペレーティングシステムが行う基本的サービスである。

プレゼンテーション層

通信する相互の計算機のデータ表現法は，必ずしも同一ではない（数値や，文字を表現するピットパターン）。そこで，計算機間のデータ表現の違いを吸収するために導入されたのが，プレゼンテーション層である。また，ネットワークを流れるデータを第三者に盗用されないために，データの暗号化を行う機能などもこの層に含まれる。

アプリケーション層

ネットワークを使用してユーザが享受できるサービスを含む応用プログラムの層である。

（2）TCP/IP モデル

TCP/IP モデルは４層からなり，ＯＳＩモデルの７層との関係を図２に示す。

ネットワークインタフェース層

ＬＡＮのプロトコル，デジタル交換網のプロトコルなど，異種のネットワークそれぞれの内部のプロトコルがこの層に属する。イーサネット，トークンリングなどさまざまなプロトコルがある。

インターネット層

この層には，異種のネットワークを越えてパケットが行き来する。下位の層のプロトコルや特性の違いをこの層で吸収する。上位の層に対して，経路を意識しないホスト間のパケット送受信のサービスを提供する。この層には，唯一インターネット・プロトコル（Ｉ

P）だけが定められている。

OSI参照モデル	TCP/IPモデル	主なプロトコル
アプリケーション層	アプリケーション層	ユーザが利用するアプリケーションごとにサービスを提供 HTTP, FTP, TELNET, SMTP,POP3
プレゼンテーション層		
セッション層		
トランスポート層	トランスポート層	TCP, UDP
ネットワーク層	インターネット層	IP
データリンク層	ネットワークインタフェース層	イーサネット, トークンリング
物理層		

図２　ＯＳＩ参照モデルと TCP/IP モデルの関係

トランスポート層

この層では，プロセス間やアプリケーションプログラム間の通信をサポートする。この層には，以下のようなプロトコルがある。

a) ＴＣＰ(Transmission Control Protocol):列車方式
コネクションを張り，バイトストリームの伝送サービスをする。
パケットの順序，エラー回復を含む信頼性を保証する。

b) ＵＤＰ(User Datagram Protocol)：宅急便方式
データグラム形式でコネクションレスの伝送サービスをする。
信頼性は保証しない。

c) ＩＣＭＰ(Internet Control Message Protocol)
インターネットワークの状態のモニタや，維持管理するための情報をホスト，ゲートウェイなどの間でやり取りするためのプロトコル。

この層のサービスを利用してアプリケーションを開発するときのインタフェースが**ソケット**である。

アプリケーション層

ＯＳＩモデルの，プレゼンテーションとアプリケーション層に対応する。

「調査項目 2」 現在，Windows や Linux ではどのようなネットワークアプリケーションが普及しているか。次の例ように，アプリケーション名と主な用途，機能を説明せよ。また，OS やプロトコル名，クライアントかサーバかも書くこと。（**6 個以上**）

アプリケーション名 （OS）	主な用途 （プロトコル）	機能 （クライアントorサーバ）
Internet Explorer (Windows)	Webページの閲覧 (HTTP)	URLを指定してWebページを表示，印刷，保存することができる JavaアプレットやJavaスクリプトなどの実行もできる (クライアント)
:	:	:

(注)Internet Explorer と Chrome, Firefox など同類のアプリはいくつ書いても 1 つとみなす。

1.2 ネットワーク上でのプロセス間通信

ソケット

プロセス間通信のためには，プロセスとプロセスの間を結ぶ経路が必要となる。通信路の端点にはソケットと呼ばれるデータ構造が割り当てられ，プロセス側からはファイルディスクリプタでアクセスすることができる。ソケットインタフェースを用いることで，プロセス間通信もファイルの入出力と同じように扱うことができる。また，ソケットを用いたプロセス間通信プログラムは，クライアントプロセスとサーバプロセスからなる。また，ネットワーク接続にはコネクション指向とコネクションレスがあるが，以下では，コネクション指向について述べる。

クライアント・サーバモデル

ネットワークアプリケーションの標準的なモデルである。**サーバ**とは，クライアントプロセスからの接続を待ち，クライアントのためにサービスを行うプロセスである。サービスを依頼する方を**クライアント**と呼び，依頼されたサービスを実行し応答を返す方をサーバと呼ぶ。

インターネットアドレス（IP アドレス）

ネットワークとコンピュータを識別するためのアドレスである。ネットワークＩＤとホストＩＤ併せて３２ビットの幅がある。インターネット上のすべてのホストは，異なった３２ビットのアドレスを持たなければならない。通常 IP アドレスは，８ビットずつ１０進表記し「．」で区切って記述する。

３２ビットのインターネットアドレスは，次の４形式のいずれかの形をしている。

クラスA，クラスB，クラスC，クラスD（マルチキャスト）

実験で用いるパソコンのアドレスは，10.139.107.XXX から 10.139.107.ZZZ までである。

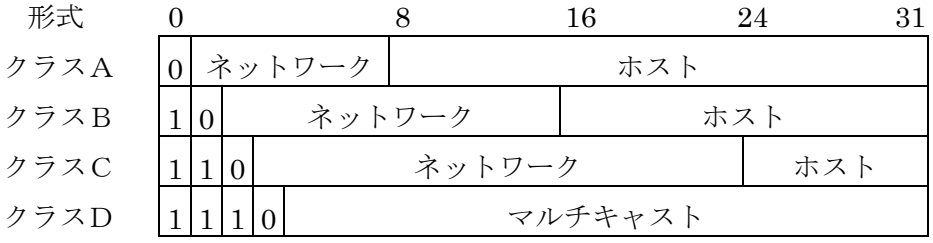


図３　ＩＰアドレスの形式

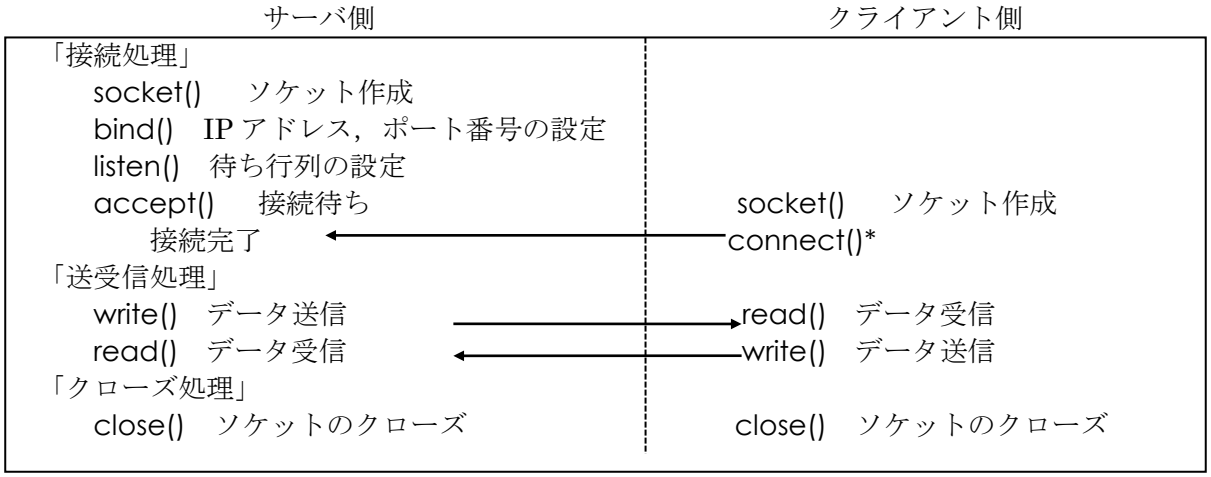
「調査項目 3」各形式の違いを調べよ。また，実験で用いているアドレスがどの形式かも理由をつけて示すこと。

ポート番号

ＴＣＰ，ＵＤＰを２つ以上のプロセスが同時に用いることが可能である。そのためには，あるデータがどのプロセスのものなのかを識別する手段が必要になる。ＴＣＰ，ＵＤＰではこの識別を１６ビットのポート番号を用いて行う。services ファイルには，サービス名とポート番号の集合が定義されている。

ポート番号 1 から 255 までは予約済みである。（4.3BSD では，1 から 1023 までスーパーユーザプロセスのために予約している。）実験では，ポート番号 20000 を用いる。

ソケットインタフェースでの通信手順



* サーバの IP アドレスと，ポート番号を指定して接続要求する。

2．実験の構成

2.1　システム構成

図４に実験を行うためのシステム構成を示す。

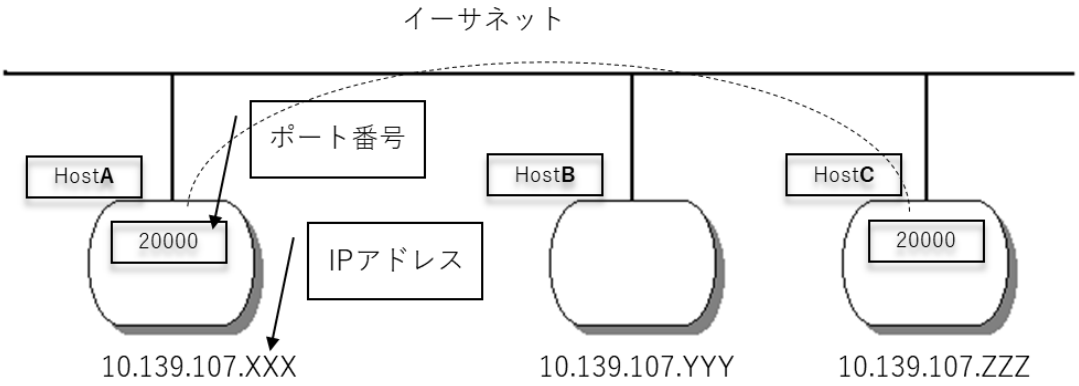


図４　実験で用いるネットワーク環境

使用機種　　DOS/V パソコン　　８台

ＯＳ　　Microsoft Windows 10 Enterprise

言語処理系その他

Visual Basic 2019

Experiment.TcpSocket.dll

実験は，イーサネットで接続されたパソコン(Host Aから Host C)を用いて行う。

プログラム開発は, Windows 上の Visual Basic で行う。Visual Basic でのプログラム開発は，イベント駆動のプログラミングスタイルとなる。

ソケット通信は，本実験のために開発した Experiment.TcpSocket.dll を用いて，ソケットベ

ースの TCP 通信を行う。

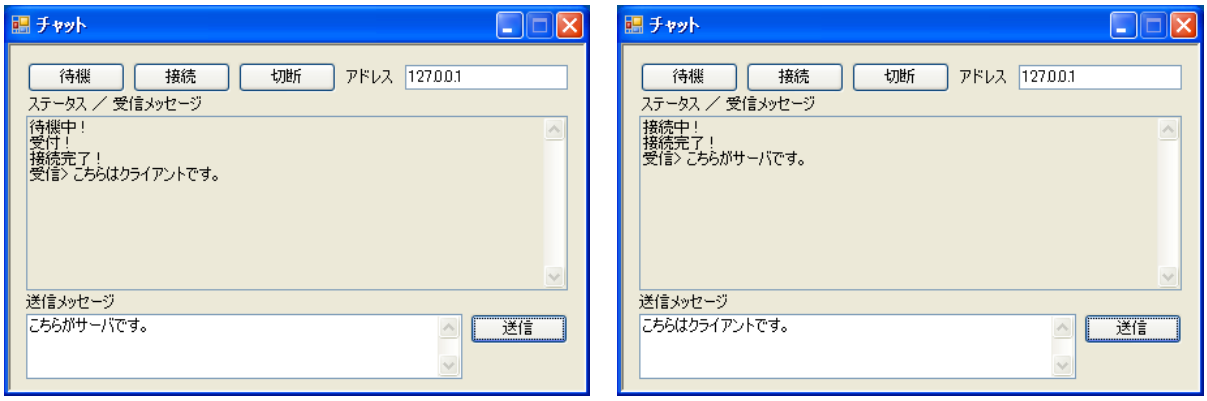
2.2 サンプルプログラム

2 つのコンピュータ間で，メッセージ送受信を行うサンプルプログラムを用意している。クライアント・サーバモデルに基づくプログラムであるが，一つのプログラムに両方の機能を持たせている。

以下に，実験で用いるサンプルプログラムファイルの一覧を示す。

Chat2019.exe	コンパイル済みサンプルプログラム
Experiment.TcpSocket.dll	実験用ソケットベース TCP 通信クラスライブラリ

サーバ側，クライアント側ともに Chat2019.exe を実行する。図 5 に実行画面を示す。



(a) サーバとして動作している画面　　(b) クライアントとして動作している画面

図 5　サンプルプログラムの動作画面

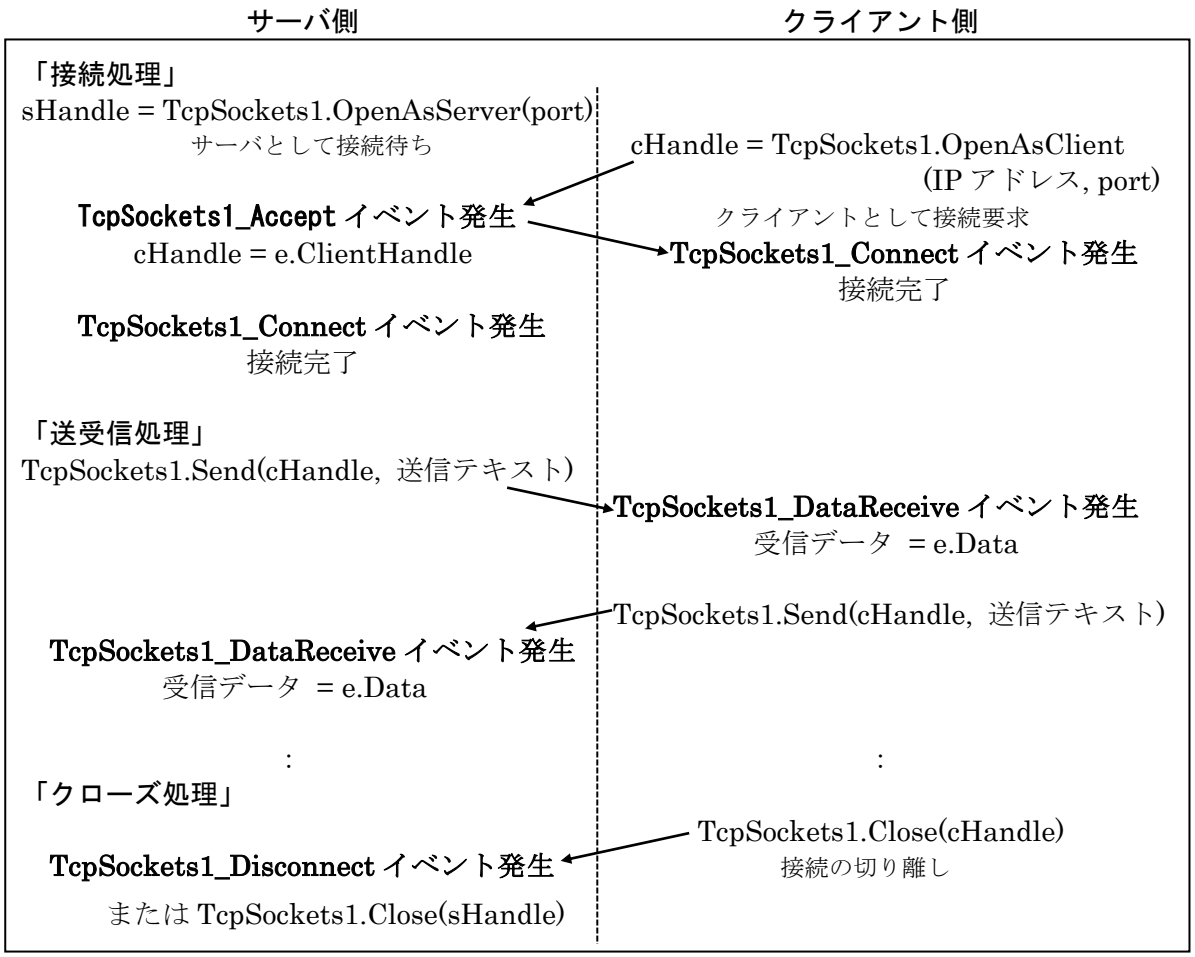
サーバは，「待機」ボタンを押すことで，20000 ポートでクライアントの接続要求を待つ（**待機中 !**）。クライアントは，アドレス欄にサーバの IP アドレスを入力し，「接続」ボタンを押すとサーバの 20000 ポートに接続要求を出す（**接続中 !**）。

接続イベントを受けたサーバは，Accept イベントが発生（受付 !）した後，Connect イベントが発生する（**接続完了 !**）。同じくクライアントにも Connect イベントが発生する(**接続完了 !**)。それぞれウィンドウ下のテキスト領域にメッセージを入力し，「送信」ボタンを押すと相手にメッセージが送信され，受信したメッセージはウィンドウ上のテキスト領域に追加される。

クライアントが「切断」ボタンを押す（**切断 !**）とサーバに切断したことが知らされる（**切断されました !**）。

以下に，Experiment.TcpSocket.dll によるソケット通信手順の概略図を示すが，詳しくは，付録の Experiment.TcpSocket 名前空間の説明を参照のこと。

Experiment.TcpSocket.dll での TCP/IP 通信手順



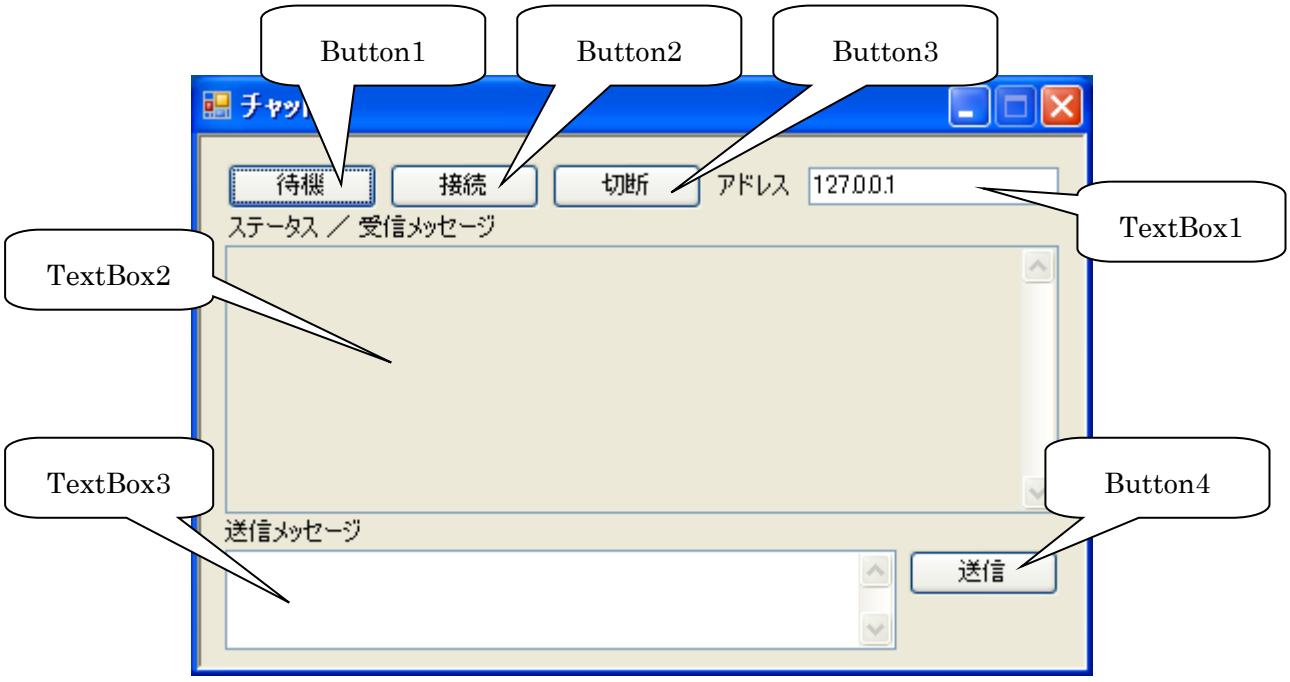


図６　画面レイアウトとオブジェクト名の対応

3．実験の手順

以下の手順に従って，実験を行う。課題」は 2 週間以内，「研究」は実験終了までに行うこと。

3.1 サンプルプログラムの変更（全員が 1 人で行うこと：2 週間）

ビルドされた実行可能プログラムは　**Chat2019¥bin¥debug¥Chat2019.exe**　にある。

「課題 1」　サンプルプログラムは，実行後に対応するボタンを押すことでサーバかクライアントとして動作する。現在の状態がサーバなのかクライアントなのかがわかるようにせよ。サーバかクライアントかを示す方法は各自で考えよ。

「課題 2」　サンプルプログラムでは，サーバは複数のクライアントと接続することができる。しかし，サーバは全クライアントからのメッセージを受信できるのに，メッセージを送信できるのは最後に接続したクライアントだけである。
　サーバが，全クライアントへメッセージを送信できるように変更せよ。
　Chat2019.exe を複数起動し，接続ボタンでクライアントにすること。
（ヒント）
　常に，最後のクライアントの Handle しか記憶していないためである。現在接続しているクライアント数と複数のクライアント Handle を記憶するようにせよ。

「課題 3」　サーバは，クライアントからの受信メッセージを，他の全クライアントに自動送信するように改良せよ。

「課題 4」　サーバからのメッセージ送信を，特定のクライアントだけに送れるようにせよ。

「研究 1」　クライアントは，自分のニックネームをサーバに知らせ，各受信メッセージが誰からの物かわかるように改良せよ。

「研究 2」　受信メッセージとともに受信した時刻を表示せよ。

「研究 3」　受信メッセージとともに送信元の IP アドレスを表示せよ。
（ヒント）
　イベントデータの e.RemoteEndPoint.Address プロパティに送信元についての情報が格納されている。ただし，ToString メソッドで文字列に変換する必要がある。

3.2 ネットワークアプリケーションの設計・開発（２人ペアで行うこと：3 週間）

オリジナルのネットワークアプリケーションを設計・開発せよ。
まず，どのような情報を送受信するかプロトコルを考えること。

例えば，三目並べゲーム[注)]の場合

情　報	送信文字列
先攻	“S”　自分が先に石を置くことを通知
後攻	”K”　相手に先に石を置いてほしいことを通知
石を置いた場所	”Gi”　i は石を置く場所に対応する数字
リセット	“R”　ゲームをはじめからやり直すことを通知

注）３行３列のゲーム盤に先手は黒石，後手は白石を置く。先に，縦・横・斜めのいずれかに３個置いた方が勝ちとなるゲーム。

４． 文字列操作関数

通信は文字列で行われる。
以下の文字列操作関数を用いて，プロトコルのコマンドと引数を分割して処理する。

関　数	機　能
Chr(charcode)	指定した文字コードに対応する文字を示す文字列型 (String) の値を返します。
Hex(number)	指定した値を 16 進数で表した文字列型 (String) を返します。
InStr([start,]string1, string2[, compare])	バリアント型 (内部処理形式 Long の Variant) の値を返します。ある文字列 (string1) の中から指定した文字列 (string2) を検索し、最初に見つかった文字位置 (先頭からその位置までの文字数) を返す文字列処理関数です。
InstrRev(string1, string2 [, start[, compare]])	ある文字列 (string1) の中から指定された文字列 (string2) を最後の文字位置から検索を開始し、最初に見つかった文字位置 (先頭からその位置までの文字数) を返す文字列処理関数です。
Microsoft.VisualBasic.Left (string, length)	バリアント型 (内部処理形式 String の Variant) の値を返します。文字列の左端から指定した文字数分の文字列を返します。
Len(string | varname)	指定した文字列の文字数または指定した変数に必要なバイト数を表す長整数型 (Long) の値を返します。
Mid(string, start[, length])	バリアント型 (内部処理形式 String の Variant) の値を返します。文字列から指定した文字数分の文字列を返します。
Replace(expression, find, replacewith [, start[, count[, compare]]])	指定された文字列の一部を、別の文字列で指定された回数分で置換した文字列を返します。
Microsoft.VisualBasic.Right (string, length)	バリアント型 (内部処理形式 String の Variant) の値を返します。文字列の右端から指定した文字数分の文字列を返します。
Str(number)	バリアント型 (内部処理形式 String の Variant) の値を返します。数式の値を文字列で表した値 (数字) で返す文字列処理関数です。
StrComp(string1, string2[, compare])	文字列比較の結果を表すバリアント型 (内部処理形式 String の Variant) の値を返します。
Val(string)	指定した文字列に含まれる数値を適切なデータ型に変換して返します。

５． 基本計画書

＜ネットワークアプリケーション基本設計書について＞
　中間レポートに，作成するアプリケーションの基本設計書をつけること。
　基本計画書には次の内容を記すこと。

基本計画書
　①アプリケーション名：開発者氏名
　②要求定義（機能設計）：システムのもつ機能
　③画面設計とヒューマンインターフェース
　④論理データ設計
　　・主要データのデータ構造
　　・プロトコル設計
　⑤開発スケジュール（分担も記すこと）

基本計画書の例

6.ネットワークアプリケーション基本設計書

6.1 アプリケーション名と概要説明

名前： メッセンジャー（開発者１氏名，開発者２氏名）

概要：手軽に楽しめるメッセンジャーソフト。身近な親友や遠くの親戚とチャットしたり、音楽や画像などのファイルを送受信したりできる上、多人数対戦型のマインスイーパーゲームまで楽しめるソフトウェアです。

6.2 要求定義 **（２０個以上）**

- ○ サーバ・クライアント構成にする。
- ○ このソフトを使い始めるときには、アカウントを作る。
- ○ アカウントにはニックネーム、パスワード、等の必要な情報を入力することで簡単に作れるようにし、アカウント情報はサーバ側に保存する。
- ○ 自分のアカウントには自分の個性が出るような画像を登録することができる。
- ○ 連絡を取りたい人を「フレンド」として登録する。
- ○ 「フレンド」は相手の ID または名前で検索し、相手の承諾が受ければ、「フレンド」に登録できる。

６． スケジュール

第 2 週：課題，研究課題のチェック
第 3 週：ネットワークアプリケーション名と概要説明（実験終了時までに）
第 4 週：作品開発
第 5 週：15:00～作品発表・相互評価（最終時限），作品アップロード

７． レポートについて

レポートには，以下のことを書くこと。

「実験の目的」 実験の目的を示せ。

「調査項目」 実験テキスト中の*「調査項目 (1 から 3)」*について，図書館や各自が持っている本などで調べ，説明すること。また，参考にした文献（著者，タイトル，出版社）も示すこと。

「課題」「研究」 ３章の実験 3.1 はどのように変更したか，アルゴリズムやデータ構造の変更点だけを示せ。
実験 3.2 については，アルゴリズムとデータ構造，プロトコル，実行(通信中の)画面などを付けて説明すること。また，工夫した点なども示せ。

「考察」 テキストの「研究項目」以外で，各自調べたことがあれば参考文献とともに示せ。(1 ページ以上)

「感想」 この実験を終えての感想，実験設備やテキストなどに対する要望があれば示せ。

7.1 レポートの構成

A レポート

1. 実験の目的
2. 調査項目 1(A4 で 1 ページ以上)
3. 調査項目 2(アプリケーションは6個以上，ただし IE, Chrome, Firefox など同類のアプリはいくつ書いても1つとみなす)
4. 調査項目 3(理由も示すこと)

6. ネットワークアプリケーション基本計画書
   - 6.1 アプリケーション名:開発者氏名
   - 6.2 要求定義(システムの持つ基本機能)
   - 6.3 開発スケジュール
     - 第 4, 5, 6, 7 週に分けて内容と分担を記すこと

B レポート

5. サンプルプログラムの改善
   - 5.1 研究 1
     - 変更・改善法の説明
   - 5.2 研究 2
     - 変更・改善法の説明
   - 5.3 研究 3
     - 変更・改善法の説明

6. (A レポートの 6.1 から 6.3 を挿入)
   - 6.4 画面設計とユーザインタフェース
     - (1) フォームの構成
     - (2) ボタン等(フォームの構成要素)の説明
   - 6.5 論理データ設計
     - (1) 主要データのデータ構造
     - (2) プロトコル設計
   - 6.6 プログラムの説明
   - 6.7 実行画面
   - 6.8 まとめ
     - (1) 問題点と今後の改善点
     - (2) 共同作業における状況(分担，スケジュール，完成度など)
7. 考察 (1 ページ以上) (あれば感想や要望も)

注意：1,2,3,4 は手書きであること。5,6 は Word などを使用して作成すること。
6 の内容は、共同研究者と同じでかまいません。
B レポートは，A レポートに付け足して，さらに番号順に並び替えて提出すること

7.2 参考文献の書き方

“著者名，タイトル，出版社，参照ページ”

7.3 プロトコル

どのような情報を通信するのか，またそれらの情報をどのようにして識別するかがわかるように書くこと。また，コマンドを用いる場合はコマンドとその意味の対応表をつけること。

サンプルプログラムリスト

```
Public Class Form1
    Const port As Integer = 20000 ' ポート番号
    Dim enc As System.Text.Encoding = System.Text.Encoding.Default ' 文字コードに「Shift-JIS」
を指定
    Dim sHandle As Long = -1 ' サーバハンドル
    Dim cHandle As Long = -1 ' クライアントハンドル

    ' 「待機」ボタンが押された
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        ' リッスンおよび接続が行われていないとき
        If sHandle = -1 AndAlso cHandle = -1 Then
            Try
                sHandle = TcpSockets1.OpenAsServer(port) ' サーバとして動作
                WriteMessage("待機中 !")
            Catch ex As Exception ' 例外がスローされたとき
                ' エラーメッセージを出力
                WriteMessage("Error: " & ex.Message)
            End Try
        End If

    End Sub

    Delegate Sub WriteMessageDelegate(ByVal str As String)

    Public Sub WriteMessage(ByVal msg As String)
        Invoke(New WriteMessageDelegate(AddressOf TextBox2.AppendText), msg & vbCrLf)
    End Sub

    ' 「接続」ボタンが押された
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        ' リッスンおよび接続が行われていないとき
        If sHandle = -1 AndAlso cHandle = -1 Then
            Try
                Dim host As String = TextBox1.Text
                ' クライアントとして動作
                cHandle = TcpSockets1.OpenAsClient(host, port)
                WriteMessage("接続中 !")
            Catch ex As Exception ' 例外がスローされたとき
                ' エラーメッセージを出力
                WriteMessage("Error: " & ex.Message)
            End Try
        End If

    End Sub

    ' 「切断」ボタンが押された
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        If sHandle <> -1 Then ' サーバとして動作しているとき
            TcpSockets1.Close(sHandle) ' リッスンをやめる
            sHandle = -1
            WriteMessage("待機終了 !")
```

```
        End If
        If cHandle <> -1 Then ' クライアントとして動作しているとき
            TcpSockets1.Close(cHandle) ' リモートとの通信を切断する
            WriteMessage("切断 !")
        End If
    End Sub

    ' 「送信」ボタンが押された
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        If cHandle <> -1 Then ' リモートに接続しているとき
            Dim msg As String = TextBox3.Text
            Dim bytes As Byte() = enc.GetBytes(msg) ' 文字列をバイト配列に変換
            Try
                TcpSockets1.Send(cHandle, bytes) ' メッセージ送信
            Catch ex As Exception ' 例外がスローされたとき
                ' エラーメッセージを出力
                WriteMessage("Error: " & ex.Message)
            End Try
        End If
    End Sub
    ' Acceptイベントが発生した
    Private Sub TcpSockets1_Accept(ByVal sender As System.Object, ByVal e As
Experiment.TcpSocket.AcceptEventArgs) Handles TcpSockets1.Accept
        ' NOTE: このロジックでは２回以上のアクセプトに対応できない

        cHandle = e.ClientHandle
        WriteMessage("受付 !")
    End Sub

    ' Connectイベントが発生した
    Private Sub TcpSockets1_Connect(ByVal sender As System.Object, ByVal e As
Experiment.TcpSocket.ConnectEventArgs) Handles TcpSockets1.Connect
        WriteMessage("接続完了 !")
    End Sub

    ' Disconnectイベントが発生した
    Private Sub TcpSockets1_Disconnect(ByVal sender As System.Object, ByVal e As
Experiment.TcpSocket.DisconnectEventArgs) Handles TcpSockets1.Disconnect
        cHandle = -1
        WriteMessage("切断されました !")
    End Sub

    ' DataReceiveイベントが発生した
    Private Sub TcpSockets1_DataReceive(ByVal sender As System.Object, ByVal e As
Experiment.TcpSocket.DataReceiveEventArgs) Handles TcpSockets1.DataReceive
        Dim msg As String = enc.GetString(e.Data) ' バイト配列を文字列に変換
        WriteMessage("受信> " & msg)
    End Sub
End Class
```

Experiment.TcpSocket 名前空間の説明

TcpSockets クラス

概要

ソケットベースの TCP 通信を提供するコンポーネント。

主なメソッド

- Public Function OpenAsServer(port As Integer) As Long

  クライアントからの接続要求の待機を開始する。

  引数１：ローカルポート

  戻り値：ハンドル

- Public Function OpenAsClient(hostname As String, _
  port As Integer) As Long

  サーバへの接続を開始する。

  引数１：リモートホスト

  引数２：リモートポート

  戻り値：ハンドル

- Public Sub Close(handle As Long)

  接続要求の待機を終了する，または，接続を切断する。

  引数１：ハンドル

- Public Function IsOpened(handle As Long) As Boolean

  接続要求を待機しているか否か，または，接続されているか否かを取得する。

  引数１：ハンドル

- Public Sub Send(handle As Long, _
  buffer As Byte())

  データを送信する。

  引数１：ハンドル

  引数２：バイト配列

- Public Sub Send(handle As Long, _
  buffer As Byte(), _
  offset As Integer, _
  size As Integer)

  データを送信する。

  引数１：ハンドル

  引数２：バイト配列

  引数３：バイト配列のオフセット

  引数４：送信するバイト数

プロパティ

- Public Property ReceiveBufferSize As Integer
  受信バッファのサイズ。
  デフォルト値：8192
- Public Property SendBufferSize As Integer
  送信バッファのサイズ。
  デフォルト値：8192
- Public SynchronizingObject As System.ComponentModel.IsynchronizeInvoke
  イベントハンドラ呼び出しのときに同期をとるコントロールのインスタンス。通常は，この TcpSockets インスタンスが属するフォームを指定する。
  デフォルト値：null 参照

イベント

- Public Event Accept As AcceptEventHandler
  新しい接続要求を受け入れたときに発生する。
- Public Event Connect As ConnectEventHandler
  リモートホストへ接続したときに発生する。
- Public Event Disconnect As DisconnectEventHandler
  リモートホストへの接続が切断したときに発生する。
- Public Event DataReceive As DataReceiveEventHandler
  リモートホストからデータを受信したときに発生する。

AccpeptEventArgs クラス

概要

Accept イベントデータ。

プロパティ

- Public ReadOnly Property ServerHandle As Long
  サーバハンドル。
- Public ReadOnly Property ClientHandle As Long
  クライアントハンドル。
- Public ReadOnly Property RemoteEndPoint As System.Net.IPEndPoint
  リモートホストのエンドポイント。

ConnectEventArgs クラス

概要

Connect イベントデータ。

プロパティ

- Public ReadOnly Property Handle As Long
  ハンドル。
- Public ReadOnly Property RemoteEndPoint As System.Net.IPEndPoint
  リモートホストのエンドポイント。

DisconnectEventArgs クラス

概要

Disconnect イベントデータ。

プロパティ

- Public ReadOnly Property Handle As Long
  ハンドル。
- Public ReadOnly Property RemoteEndPoint As System.Net.IPEndPoint
  リモートホストのエンドポイント。

DataReceiveEventArgs クラス

概要

DataReceive イベントデータ。

プロパティ

- Public ReadOnly Property Handle As Long
  ハンドル。
- Public ReadOnly Property RemoteEndPoint As System.Net.IPEndPoint
  リモートホストのエンドポイント
- Public ReadOnly Property Byte()
  受信データ。

更新履歴

1996.03.29　初版
　　開発環境 Visual Basic Ver.2.0，EZWsock.VBX Ver.0.2
1997.04.03
　　課題とヒントを加筆
2001.05.24
　　開発環境を Visual Basic Ver.6.0，socketc.ocx に変更
2002.05.07
　　サンプルプログラムを修正
2005.03.17
　　レポートについて説明を加筆
2008.02.01
　　字句の訂正
2009.12.01
　　開発言語を VB6 から VB2005，Experiment.TcpSocket.dll に変更
2012.01.25
　　開発言語を VB2005 から VB2010 に変更
2013.11.20
　　基本計画書の説明を追加
2014.12.10
　　用語の統一
2015.11.11
　　レポートの注意事項を変更

2019.02.06

　　開発言語を VB2010 から VB2013 に変更
2020.11.26
　　開発言語を VB2013 から VB2015 に変更
2021.03.29
　　開発言語を VB2015 から VB2019 に変更
　　実験時間を 3 時間 7 週から 4 時間 5 週に変更

★　実験環境の構築手順

Visual Studio の設定

1. %SYSTEMROOT%¥Microsoft.NET¥Framework に Experiment.TcpSocket.dll と Experiment.TcpSocket.xml（IntelliSense 用）をコピーする。
2. Visual Studio から任意のフォームアプリケーションプロジェクト（新規でも可）を開いて，ツールボックスの全般タブ（もしくは新規タブを作る）にて，アイテムの選択を開く（ロードに時間が掛かる場合あり）。
3. 「.NET Framework コンポーネント」に上のパスから DLL を追加し，チェックを付ける．

これで，永続的にコンポーネントが登録される。

プロジェクトの設定

1. 「My Project」またはプロジェクトのプロパティの参照タブにて，参照設定に「Experiment.TcpSocket」を追加する。
2. 「ローカルにコピーする」をプロパティウィンドウから True にする（重要!!）。
3. 「インポートされた名前空間」で，Experiment.TcpSocket にチェックを付ける。
4. フォームにコンポーネントを貼り付ける。
5. コンポーネントのプロパティの SynchronizingObject をフォームにする（重要!!）。

これで，他の環境にコピーしても同じように動く。