



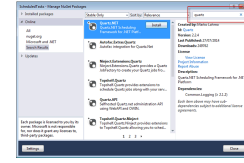
**MQL là gì? Những tính năng mà MQL4 và MQL5 sở hữu?**

07/04/2023



**Top 10 câu hỏi phỏng vấn Linux hàng đầu dành cho Ứ...**

04/04/2023



**Tự động thực thi lịch trong ASP.NET**

14/01/2023

# C# – Lập trình Socket giao tiếp TCP client/server

Hoài Nam

10/12/2020



## C# – Lập trình Socket giao tiếp TCP client/server



Trong lập trình, Socket là một API (Application Programming Interface) cung cấp các phương thức để giao tiếp thông qua mạng.

Trước khi bắt đầu tìm hiểu và viết một ví dụ đơn giản về socket, bạn có thể tham khảo bài viết “[Networking – Một số khái niệm cơ bản](#)” để có cái nhìn sơ lược về những khái niệm cơ bản trong lập trình mạng.

## Các lớp .Net cơ bản trong lập trình mạng

Các lớp này được cung cấp trong hai namespace [System.Net](#) và [System.Net.Sockets](#). Hai namespace này chứa rất nhiều lớp dùng trong lập trình mạng, nhưng trong phạm vi bài viết ta chỉ quan tâm đến các lớp sau::

Class	Namespace	Description
<a href="#">IPAddress</a>	<a href="#">System.Net</a>	Provides an Internet Protocol (IP) address.
<a href="#">IPEndPoint</a>	<a href="#">System.Net</a>	Represents a network endpoint as an IP address and a port number.
<a href="#">TcpListener</a>	<a href="#">System.Net.Sockets</a>	Listens for connections from TCP network clients.
<a href="#">Socket</a>	<a href="#">System.Net.Sockets</a>	Implements the <a href="#">Berkeley sockets</a> interface.
<a href="#">TcpClient</a>	<a href="#">System.Net.Sockets</a>	Provides client connections for TCP network services.
<a href="#">NetworkStream</a>	<a href="#">System.Net.Sockets</a>	Provides the underlying stream of data for network access.

## Kết nối Server-Client với TCP/IP

Khi được chạy, server cần được xác định rõ địa chỉ IP và sẽ “lắng nghe” trên một port cụ thể. Server sẽ nằm trong trạng thái này cho đến khi client gửi đến một yêu cầu kết nối. Sau khi được server chấp nhận, một connection sẽ hình thành cho phép server và client giao tiếp với nhau.

Cụ thể hơn, các bước tiến hành trên server và client mà ta cần thực hiện sử dụng giao thức TCP/IP trong C# (có thể chạy server và client trên cùng một máy):

### Server:

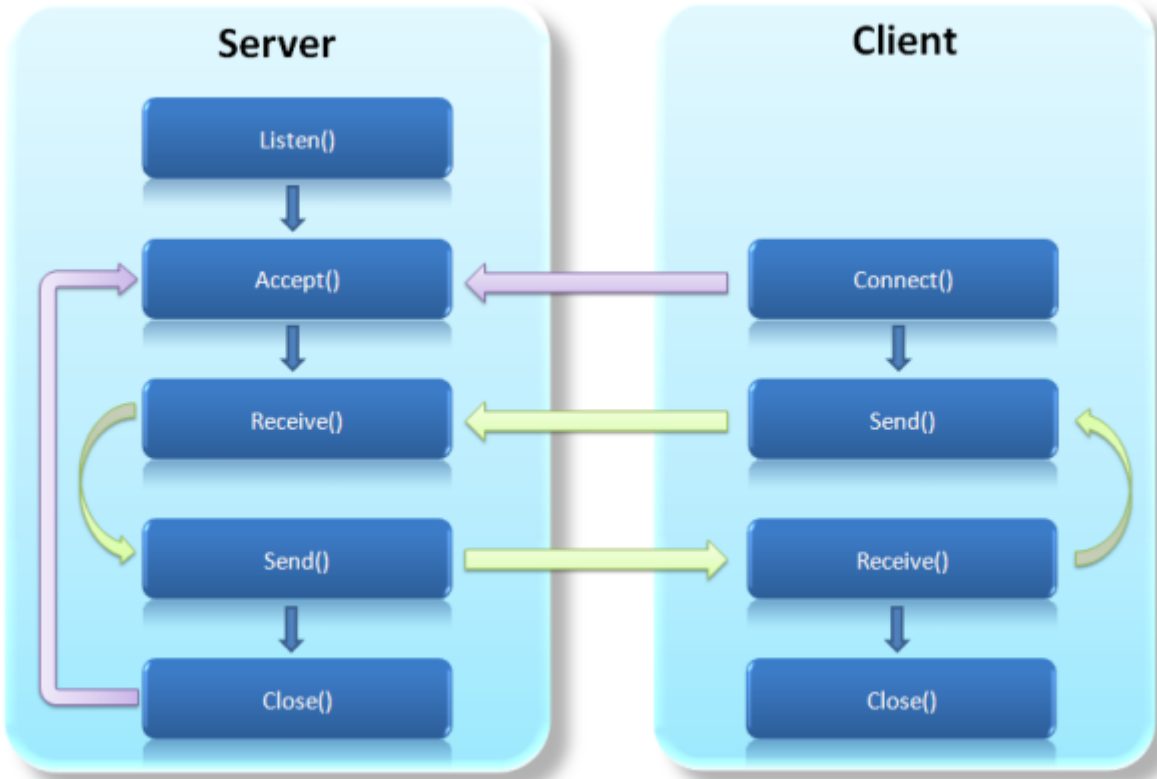
1. Tạo một đối tượng [System.Net.Sockets.TcpListener](#) để bắt đầu “lắng nghe” trên một cổng cục bộ.
2. Đợi và chấp nhận kết nối từ client với phương thức `AcceptSocket()`. Phương thức này trả về một đối tượng [System.Net.Sockets.Socket](#) dùng để gửi và nhận dữ liệu.
3. Thực hiện giao tiếp với client.
4. Đóng Socket.

Thông thường quy trình này sẽ được đặt trong một vòng lặp (lặp lại bước 2) để chấp nhận nhiều kết nối cùng lúc (sử dụng Thread) hoặc các kết nối lần lượt.

### Client:

1. Tạo một đối tượng [System.Net.Sockets.TcpClient](#)
2. Kết nối đến server với địa chỉ và port xác định với phương thức `TcpClient.Connect()`
3. Lấy luồng (stream) giao tiếp bằng phương thức `TcpClient.GetStream()`.
4. Thực hiện giao tiếp với server.
5. Đóng luồng và socket.

Quy trình này có thể được minh họa theo mô hình sau:



## Example v1: Gửi nhận dữ liệu dạng byte[]

Lớp `NetworkStream` và `Socket` cung cấp các phương thức gửi và nhận dữ liệu dạng mảng byte. Vì vậy bạn cần phải thực hiện các bước chuyển đổi dữ liệu sang dạng byte và ngược lại. Trong ví dụ sau tôi sử dụng dữ liệu dạng văn bản ASCII trong console, và dùng các lớp trong namespace `System.Text` để chuyển đổi. Có hai cách bạn có thể áp dụng:

- Dùng các static property của lớp abstract `System.Text.Encoding` với các phương thức `GetString()` và `GetBytes()`.
- Tạo đối tượng có kiểu `XXXEncoding` (thừa kế từ `System.Text.Encoding`). Ví dụ: `UTF8Encoding`, `ASCIIEncoding`,...

Một ví dụ gửi nhận dữ liệu đơn giản nhất sử dụng `TCPListener`, `Socket` (phía server) và `TCPClient`, `NetworkStream` (phía client) dạng mảng byte với địa chỉ loop-back `127.0.0.1` trên cùng một máy.

Tạo hai dự án console là `Y2Server` và `Y2Client` với nội dung sau:

**Y2Server.cs (v1):**

```
using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

public class Y2Server {

    private const int BUFFER_SIZE=1024;

    private const int PORT_NUMBER=9999;

    static ASCIIEncoding encoding=new ASCIIEncoding();

    public static void Main() {

        try {

            IPAddress address = IPAddress.Parse("127.0.0.1");

            TcpListener listener=new TcpListener(address,PORT_NUMBER);

            // 1. listen

            listener.Start();

            Console.WriteLine("Server started on "+listener.LocalEndPoint);

            Console.WriteLine("Waiting for a connection...");

            Socket socket=listener.AcceptSocket();

            Console.WriteLine("Connection received from " + socket.RemoteEndPoint);

            // 2. receive

            byte[] data=new byte[BUFFER_SIZE];

            socket.Receive(data);

            string str=encoding.GetString(data);

            // 3. send

            socket.Send(encoding.GetBytes("Hello "+str));

            // 4. close

            socket.Close();

            listener.Stop();
        }
        catch (Exception ex) {

            Console.WriteLine("Error: " + ex);
        }
    }
}
```

```
        Console.Read();  
    }  
}
```

**Y2Client.cs (v1):**

```
using System;

using System.IO;

using System.Net;

using System.Text;

using System.Net.Sockets;

public class Y2Client{

    private const int BUFFER_SIZE=1024;

    private const int PORT_NUMBER=9999;

    static ASCIIEncoding encoding= new ASCIIEncoding();

    public static void Main() {

        try {

            TcpClient client = new TcpClient();

            // 1. connect

            client.Connect("127.0.0.1",PORT_NUMBER);

            Stream stream = client.GetStream();

            Console.WriteLine("Connected to Y2Server.");

            Console.Write("Enter your name: ");

            string str = Console.ReadLine();

            // 2. send

            byte[] data=encoding.GetBytes(str);

            stream.Write(data,0,data.Length);

            // 3. receive

            data =new byte[BUFFER_SIZE];

            stream.Read(data,0,BUFFER_SIZE);

            Console.WriteLine(encoding.GetString(data));

            // 4. Close
            stream.Close();

            client.Close();

        }

        catch (Exception ex) {
```

```
        Console.WriteLine("Error: " + ex);
    }

    Console.Read();
}
}
```

Để kiểm tra ví dụ, bạn chạy server trước, cửa sổ console của server sẽ hiển thị:

Server started on 127.0.0.1:9999

Waiting for a connection...

Tiếp đến cho chạy client, nếu kết nối thành công, server sẽ hiển thị thêm dòng thông báo tương tự như sau:

Connection received from 127.0.0.1:2578

Chuyển qua cửa sổ console của client và nhập tên của bạn vào, nếu nhận được dữ liệu, server sẽ gửi trả lại dòng thông điệp “Hello [Your Name]”

Connected to Y2Server.

Enter your name: Yin Yang

Hello Yin Yang

Ngay sau bước này, cả server và client đều thực hiện đóng kết nối.

## Example v2: Sử dụng StreamReader và StreamWriter

Sẽ tiện lợi hơn nếu ta sử dụng StreamReader và StreamWriter để gửi nhận dữ liệu mà không cần bước chuyển đổi qua lại mảng byte. Các đối tượng StreamReader và StreamWriter có thể được khởi tạo trực tiếp từ NetworkStream. Thuộc tính AutoFlush của StreamWriter thường được đặt là true để tự động gửi dữ liệu mà không cần đợi bộ đệm đầy hoặc bạn phải gọi thủ công phương thức Flush().

Ví dụ sau sử dụng vòng lặp để thực hiện gửi nhận dữ liệu liên tục giữa server/client cho đến khi client nhập vào chuỗi “exit”:

**Y2Server.cs (v2):**

```
using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

public class Y2Server {

    private const int BUFFER_SIZE=1024;

    private const int PORT_NUMBER=9999;

    static ASCIIEncoding encoding=new ASCIIEncoding();

    public static void Main() {

        try {

            IPAddress address = IPAddress.Parse("127.0.0.1");

            TcpListener listener=new TcpListener(address,PORT_NUMBER);

            // 1. listen

            listener.Start();

            Console.WriteLine("Server started on "+listener.LocalEndPoint);

            Console.WriteLine("Waiting for a connection...");

            Socket socket=listener.AcceptSocket();

            Console.WriteLine("Connection received from " + socket.RemoteEndPoint);

            var stream = new NetworkStream(socket);

            var reader=new StreamReader(stream);

            var writer=new StreamWriter(stream);

            writer.AutoFlush=true;

            while(true)

            {

                // 2. receive

                string str=reader.ReadLine();

                if(str.ToUpper()=="EXIT")

                {

                    writer.WriteLine("bye");

                    break;

                }

                // 3. send

                writer.WriteLine("Hello "+str);

            }

        }

    }

}
```

```
        // 4. close

        stream.Close();

        socket.Close();

        listener.Stop();

    }
    catch (Exception ex) {
        Console.WriteLine("Error: " + ex);
    }

    Console.Read();
}
```

Y2Client.cs (v2):



```
using System;

using System.IO;

using System.Net;

using System.Text;

using System.Net.Sockets;

public class Y2Client{

    private const int BUFFER_SIZE=1024;

    private const int PORT_NUMBER=9999;

    static ASCIIEncoding encoding= new ASCIIEncoding();

    public static void Main() {

        try {

            TcpClient client = new TcpClient();

            // 1. connect

            client.Connect("127.0.0.1",PORT_NUMBER);

            Stream stream = client.GetStream();

            Console.WriteLine("Connected to Y2Server.");

            while(true)

            {

                Console.Write("Enter your name: ");

                string str = Console.ReadLine();

                var reader=new StreamReader(stream);

                var writer=new StreamWriter(stream);

                writer.AutoFlush=true;

                // 2. send

                writer.WriteLine(str);

                // 3. receive

                str=reader.ReadLine();

                Console.WriteLine(str);

                if(str.ToUpper()=="BYE")

                    break;

            }

        }

    }

}
```

```
    }

    // 4. close

    stream.Close();

    client.Close();

}

catch (Exception ex) {

    Console.WriteLine("Error: " + ex);

}

Console.Read();

}
```

Bạn chạy ví dụ này giống như ví dụ đầu tiên và gõ ‘exit’ vào client để thoát ứng dụng.





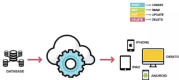


[Bài trước](#)  
**[Bắt đầu học PHP với chương trình "Hello World"](#)**

[Bài sau](#)

**[Mật khẩu mạnh với PBKDF2 + HMACSHA1 bằng cách sử dụng Rfc2898DeriveBytes](#)**



PHỔ BIẾN	MỚI NHẤT
<div></div> <div>ASP.NET MVC 4 AllowAnonymous Attribute and Authorize Attribute</div> <div>29/01/2020</div>	
<div></div> <div>JWT - Từ cơ bản đến chi tiết</div> <div>06/02/2020</div>	
<div></div> <div>Lập trình hướng đối tượng (OOP) trong C#</div> <div>29/01/2020</div>	
<div></div> <div>Giới thiệu về Flutter - Một SDK cross-platform dành cho mobile app của Google</div> <div>29/02/2020</div>	
<div></div> <div>RESTful API là gì ?</div> <div>18/06/2021</div>	



Mật khẩu mạnh với PBKDF2 + HMACSHA1 bằng cách sử dụng Rfc2898DeriveBytes trong C#

10/12/2020



7 nguyên tắc quân sự có thể áp dụng vào chiến lược marketing

12/12/2020



OOP trong C#: Tính trừu tượng trong C#

27/02/2021



Tổng hợp 1000 bài tập C

01/06/2021



Flutter vs React Native - Những điều bạn cần biết

01/03/2020



10 PHONG CÁCH SÔNG TRÊN THẾ GIỚI (PHẦN 2)

23/01/2020



Tuổi trẻ rồi sẽ qua, tuổi già rồi sẽ đến

05/02/2020



JSON Web Tokens (JWT) vs Sessions

31/12/2019




ASP.NET Core - CRUD With React.js And Entity Framework Core

06/02/2020



JWT in ASP.NET Core

29/01/2020



## JSON Web Token là gì? Token-based authentication là gì?

05/02/2020

### Thẻ loại

[Coding \(31\)](#)

[Chuyện bên lề \(18\)](#)

[Network Security \(4\)](#)

[Nghề nghiệp \(2\)](#)

### Tags

[JWT](#)

[OOP](#)

[Flutter](#)

[ENTITY FRAMEWORK](#)

[Csharp](#)

[Phong cách sống](#)

[Tuổi trẻ](#)

[JSON Web Token](#)

[ASP.NET Core](#)

[MVC](#)

[Sói](#)

[Kỹ năng sống](#)

[An ninh mạng](#)

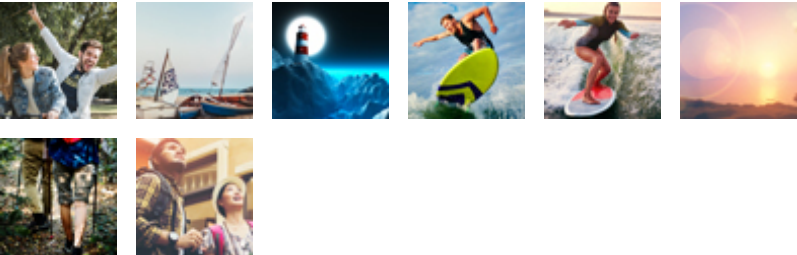
[Lịch vạn niên](#)

[Giao thừa](#)

Giới thiệu

Website phần lớn là các bài viết chia sẻ những kiến thức trong lĩnh vực công nghệ nói chung và trong ngành lập trình nói riêng. Bên cạnh đó có cả các bài viết chia sẻ những kiến thức bổ ích, những điều thú vị trong cuộc sống.

Instagram



Liên hệ

Đăng ký email để nhận bài viết từ chúng tôi

EMAIL ADDRESS

→

Theo dõi chúng tôi

Mạng xã hội



Lượt truy cập:

135686