ξ4. Lập trình socket

Xây dựng ứng dụng dùng TCP stream socket

Ứng dụng xây dựng luồng byte TCP hai chiều tuần tự và tin cậy. Ứng dụng gồm hai bộ phận, chạy trên server và chạy trên client. Ứng dụng server cần chạy trước để nghe và đón nhận kết nối từ client. Trong ví dụ này, server được xây dựng theo dạng đồng bộ. Có nghĩa rằng, hoạt động của thread sẽ bị chặn cho đến khi kết nối được server chấp nhận. Client tự đóng kết nối khi nó gửi thông điệp <TheEnd> cho server.

Chương trình SocketServer.cs:

```
using System;
using System.Net.Sockets;
using System.Net;
using System.Text;

public class SocketServer
{
   public static void Main(string [] args)
   {
   // Thiết lập điểm cuối cục bộ cho socket
      IPHostEntry ipHost = Dns.Resolve("localhost");
      IPAddress ipAddr = ipHost.AddressList[0];
      IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, 10000);

   // Tạo mới socket Tcp/Ip
      Socket sListener = new Socket(AddressFamily.InterNetwork,
                                    SocketType.Stream, ProtocolType.Tcp);
   // Ràng buộc socket vừa tạo với điểm cuối và lắng nghe kết nối đến
      try
      {
         sListener.Bind(ipEndPoint);
         sListener.Listen(10);

         // Bắt đầu lắng nghe kết nối đến
         while (true)
         {
            Console.WriteLine("Dang doi ket noi tren cong {0}",ipEndPoint);

            // chương trình bị trì hoãn trong khi chờ kết nối đến
            Socket handler = sListener.Accept();
            string data = null;

            // có client đang cố gắng truy xuất đến
               while(true)
               {
                  byte[] bytes = new byte[1024];
```

```
                int bytesRec = handler.Receive(bytes);
                data += Encoding.ASCII.GetString(bytes, 0, bytesRec);
                if (data.IndexOf("<TheEnd>") > -1) break;
            }

            // In nội dung dữ liệu ra màn hình
            Console.WriteLine("Van ban da nhan: {0}",data);
            string theReply = "Nhan duoc " + data.Length.ToString()
                              + " ky tu...";
            byte[] msg = Encoding.ASCII.GetBytes(theReply);
            handler.Send(msg);
            handler.Shutdown(SocketShutdown.Both);
            handler.Close();
         }
      }
      catch(Exception e)
      {
         Console.WriteLine(e.ToString());
      }
   }
}
```

Bước đầu tiên trong hoạt động của ứng dụng là tạo ra điểm kết thúc cục bộ. Trước khi tạo socket để nó nghe và đón nhận kết nối, nhất thiết phải có địa chỉ điểm kết thúc cục bộ cho socket (local endpoint). Điểm kết thúc là sự kết hợp địa chỉ IP với số hiệu cổng.

Lớp Dns cung cấp các thông tin địa chỉ. Khi một máy tính có nhiều hơn một địa chỉ mạng, hoặc do máy tính có nhiều hơn một card mạng, lớp Dns trả về tất cả các địa chỉ, và ứng dụng phải chọn lấy một trong số đó. Ở đây ta tạo IPEndPoint cho server bằng cách kết hợp địa chỉ IP đầu tiên được trả về bởi Dns.Resolve() với địa chỉ cổng 10000.

Tiếp theo, ta sinh một stream socket bằng cách tạo bản sao kế thừa lớp Socket.

Phép liệt kê AddressFamily biểu thị hình thức địa chỉ mà một bản sao Socket có thể dùng để phân giải địa chỉ. Một số tham số quan trọng được nêu sau đây.

Giá trị của AddressFamily	Mô tả
InterNetwork	Địa chỉ IPv4.
InterNetworkV6	Địa chỉ IPv6.
Ipx	Địa chỉ IPX hoặc SPX.
NetBios	Địa chỉ NetBios.

Tham số SocketType để nhằm phân biệt giữa một socket TCP với một socket UDP. Các giá trị khác có thể như sau:

Giá trị của SocketType	Mô tả
Dgram	Hỗ trợ các datagram. Giá trị Dgram đòi hỏi phải kết hợp với giá trị cuả ProtocolType là Udp và giá trị của AddressFamily là InterNetwork.
Raw	Hỗ trợ truy xuất tới hạ tầng giao thức tầng chuyển vận.
Stream	Hỗ trợ các stream socket. Giá trị Stream phải được kết hợp với giá trị ProtocolType là Tcp và giá trị InterNetwork của AddressFamily.

Tham số cuối cùng xác định kiểu giao thức, nó yêu cầu giao thức cho socket. Các giá trị quan trọng cho tham số ProtocolType là:

Giá trị của ProtocolType	Mô tả
Raw	Giao thức gói tin kiểu Raw.
Tcp	Giao thức TCP.
Udp	Giao thức TCP.
IP	Giao thức IP.

Bước tiếp theo là đặt tên socket với phương thức Bind(). Khi socket mở ra bởi hàm tạo, socket chưa được gán tên, nhưng nó đã có sẵn bộ mô tả. Để socket của client có thể nhận diện được stream socket TCP của server, nó phải có tên gọi. Phương thức Bind() ràng buộc một socket với điểm kết thúc cục bộ. Bind() phải được gọi thi hành trước khi gọi Listen() hoặc Accept().

Giờ là lúc có thể lắng nghe các kết nối đến với phương thức Listen(). Giá trị 10 trong ví dụ chỉ rằng, socket chấp nhận tối đa 10 kết nối đợi phục vụ trong hàng đợi của nó.

Khi socket của server đã ở trạng thái lắng nghe thì bước tiếp theo là chấp nhận kết nối đến bằng Accept(). Phương thức Accept() được dùng để nhận kết nối từ client, nó chặn thread chương trình gọi cho đến khi có một kết nối đến.

Phương thức Accept() trích lấy yêu cầu kết nối đầu tiên trong hàng đợi và tạo một socket mới để xử lý yêu cầu. Dù tạo ra socket mới thì socket cũ vẫn tiếp tục lắng nghe và thế, nó có thể dùng kỹ thuật multithread để chấp nhận thêm các kết nối khác từ client.

Khi client và server đã kết nối với nhau, các phương thức Send() và Receive() của lớp Socket có thể dùng để trao đổi dữ liệu. Phương thức Receive() nhận lấy dữ liệu từ socket và ghi vào mảng byte chỉ ra dưới dạng tham số trong câu lệnh. Trong ví dụ trên, ta tìm kiếm các ký tự kết thúc thông điệp truyền đến. Nếu không thấy, thì đoạn mã lại lặp lại để tiếp tục lắng nghe dữ liệu đến. Nếu đã thấy, nó hiện nội dung dữ liệu nhận được lên màn hình.

Thoát khỏi cấu trúc lặp, ta chuẩn bị một mảng byte mới, msg, chứa lời đáp cho client. Lời đáp này được gửi đi bằng phương thức Send().

Để chắc chắn không còn sót dữ liệu trong bộ đệm, ta gọi Shutdown() trước khi gọi Close(). SocketShutdown có thể là một trong ba giá trị khác nhau áp dụng cho socket:

Giá trị của ProtocolShutdown	Mô tả
Both	Đóng socket cả hai hướng truyền và nhận.
Receive	Đóng socket theo hướng nhận.
Send	Đóng socket theo hướng truyền.

Chương trình SocketClient.cs:

Ứng dụng client giống ứng dụng server ở chỗ tạo điểm kết thúc cục bộ, tạo socket kế thừa, gửi hay nhận dữ liệu và đóng socket.

```
using System;
using Systern.Net.Sockets;
using Systern.Net;
using Systern.Text;

public class SocketClient
{
   public static void Main(string [] args)
   {

    // bộ đệm chứa dữ liệu gửi đến
    byte[] bytes = new byte[1024];
```

```
    // kết nối đến server
    try
    {
       // cho socket gắn với điểm kết thúc tại server
       IPHostEntry ipHost = Dns.Resolve("127.0.0.1");
       IPAddress ipAddr = ipHost.AddressList[0];
       IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, 10000);

       Socket sender = new Socket(AddressFamily.Internetwork,
                                  SocketType.Stream, ProtocolType.Tcp);

       // nối socket với điểm kết thúc tại server

          sender.Connect(ipEndPoint);

          Console.WriteLine("Socket noi den {0}",
          sender.RemoteEndPoint.ToString());

          string theMessage = "This is a test";

          byte[] msg = Encoding.ASCII.GetBytes(theMessage+"<TheEnd>");

          // Gửi dữ liệu qua socket
          int bytesSent = sender.Send(msg);

          // Nhận đáp trả từ server
          int bytesRec = sender.Receive(bytes);

          Console.WriteLine("Du lieu tu server: {0}",
                            Encoding.ASCII.GetString(bytes,0, bytesRec));

          // Giải phóng socket
          sender.Shutdown(SocketShutdown.Both);
          sender.Close();

    }
    catch(Exception e)
    {
       Console.WriteLine("Exception: {0}", e.ToString());
    }
  }
}
```

Chỉ có thêm một phương thức mới là Connect(), được dùng để kết nối với server.