ξ4. Lập trình socket

# Lập trình socket bất đồng bộ (tiếp)

**Ứng dụng server bất đồng bộ**

Sơ đồ sau mô tả hoạt động bất đồng bộ trên server.

Đóng Socket

Đợi sự kiện

Đợi sự kiện

Bắt đầu lắng nghe

Tạo Socket: Listener Socket

Ràng buộc socket

Yêu cầu khởi tạo kết nối

Gửi yêu cầu truyền: BeginSend

AcceptCallback(): ClientHandler Socket

Gửi yêu cầu nhận: BeginReceive

ReceiveCallback()

Đợi sự kiện

Thiết lập sự kiện

SendCallback()

**Thread gốc**

**Thread thứ hai**

Ứng dụng Server bắt đầu bằng tiến trình tạo socket, sau đó vào trạng thái lắng nghe cổng được ràng buộc. Ứng dụng server bất đồng bộ đòi hỏi phương thức BeginAccept() chấp nhận kết nối bất đồng bộ. Và, sau khi kết nối đã được thiết lập, nó sử dụng phương thức BeginReceive() và BeginSend() để truyền và nhận với socket của client.

Sau khi đã gọi BeginAccept(), ta lập ra một sự kiện phát sinh trong trạng thái đợi, theo đó, một thead khác của ứng dụng có thể tiếp tục thi hành ngay khi một client cố kết nối đến. Nếu không làm thế, ứng dụng sẽ kết thúc trước lúc client có thể làm gì, bởi vì bản chất bất đồng bộ của server là như vậy. Lớp ManualResetEvent được dùng ở đây để lập và đợi các sự kiện:

AsyncCallback aCallback = new AsyncCallback(AcceptCallback);

// sListener là một bản sao kế thừa của lớp Socket

sListener.BeginAccept(aCallback, sListener);

// giả thiết: socketEvent là bản sao kế thừa của ManualResetEvent()

socketEvent.WaitOne();

Phương thức BeginAccept() lấy hai tham số: một hàm delegate bất đồng bộ mà nó tham chiếu tới phương thức gọi ngược; và, một tham số dùng để đẩy dữ liệu cho hàm gọi ngược (ở ví dụ cụ thể này là chính bản thân listener socket). Phương thức BeginAccept() gọi hàm AcceptCallback() khi nhận được một kết nối mới ở socket. Hàm gọi ngược này trả về socket mới cho client.

Kỹ thuật lập trình socket bất đồng bộ dùng các thread khác nhau cho các hành động bất đồng bộ khác nhau. Thread chính có trách nhiệm khởi động socket listener, một thread khác thực hiện chấp nhận các yêu cầu kết nối mới, một thread nữa có trách nhiệm nhận và gửi dữ liệu. Có thể lấy ID của thread hiện hành bằng hàm GetCurrentThreadID() thuộc lớp AppDomain.

Console.WriteLine("Main Thread ID: " + AppDomain.GetCurrentThreadId());

Lưu ý, ta không cần tự khởi chạy các thread khác nhau một cách tường minh, .NET Framework sẽ tự động hỗ trợ.

Sau khi chấp nhận kết nối bằng cách gọi phương thức EndAccept(), socket mới có thể liên lạc với client bằng cách gọi các phương thức bất đồng bộ BeginReceive() và BeginSend():

Socket handler = listener.EndAccept(ar);

handler.BeginReceive(buffer, 0, buffer.Length, 0, new

AsyncCallback(ReceiveCallback), handler);

Trước tiên, hàm ReceiveCallback() gọi EndReceive() để hoàn thành việc xử lý tác vụ bất đồng bộ đang thi hành. Sau đó, ta tìm ký tự hay dấu hiệu kết thúc thông điệp. Khi thấy, ta dùng BeginSend() để gửi dữ liệu cho client; nếu chưa thấy, ta gọi BeginReceive() để tiếp nhận phần dữ liệu còn lại:

int bytesRead = handler.EndReceive(ar);

if (bytesRead > 0)

{

content += Encoding.ASCII.GetString(buffer, 0, bytesRead);

if (content.IndexOf(".") > -1)

{

Console.WriteLine("Read {0} bytes from socket. \n Data: {1}",

content.Length, content);

byte[] byteData = Encoding.ASCII.GetBytes(content);

handler.BeginSend(byteData, 0 , byteData.Length, 0 ,

new AsyncCallback(SendCallback), handler);

}

else

{

handler.BeginReceive(buffer, 0 , buffer.Length, 0,

new AsyncCallback(ReceiveCallback), handler);

}

}

Phương thức SendCallback() kết thúc hoạt động bằng cách gọi EndSend(), đóng mã quản lý socket (handler socket), và thiết lập ManualResetEvent ở trong thread chính để ứng dụng có thể tiếp tục tiến hành:

int bytesSent = handler.EndSend(ar);

Console.WriteLine("Sent {0} bytes to Client.", bytesSent);

handler.Shutdown(SocketShutdown.Both);

handler.Close();

socketEvent.Set();

Mã đầy đủ của ứng dụng client, AsyncServer.cs, như sau:

using System;

using System.Net.Sockets;

using System.Net;

using System.Text;

using System.Threading;

public class AsyncServer

{

// Bộ đệm để truyền nhận dữ liệu

public static byte[] buffer = new byte[1024];

// Lớp sự kiện để hỗ trợ truyền nhận đồng bộ

public static ManualResetEvent socketEvent = new ManualResetEvent(false);

public static void Main(string [] args)

{

Console.WriteLine("Main Thread ID:" + AppDomain.GetCurrentThreadld());

byte[] bytes = new byte[1024];

IPHostEntry ipHost = Dns.Resolve(Dns.GetHostName());

IPAddress ipAddr = ipHost.AddressList[0];

IPEndPoint localEnd = new IPEndPoint(ipAddr, 10000);

Socket sListener = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

// thi hành ràng buộc socket

sListener.Bind(localEnd);

// bắt đầu lắng nghe

sListener.Listen(10);

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

AsyncCallback aCallback = new AsyncCallback(AcceptCallback);

// Chức năng chấp nhận kết nối bất đồng bộ

sListener.BeginAccept(aCallback,sListener);

// đợi các thread khác hoàn thành

socketEvent.WaitOne();

}

public static void AcceptCallback(IAsyncResult ar)

{

Console.WriteLine("AcceptCallback Thread ID:" +

AppDomain.GetCurrentThreadId());

Socket listener = (Socket)ar.Async State;

// Socket mới

Socket handler = listener.EndAccept(ar);

handler.BeginReceive(buffer, 0 , buffer.Length, 0,

new AsyncCallback(ReceiveCallback) , handler);

}

public static void ReceiveCallback(IAsyncResult ar)

{

Console.WriteLine("ReceiveCallback Thread ID:" +

AppDomain.GetCurrentThreadld());

string content = String.Empty;

Socket handler = (Socket) ar.AsyncState;

int bytesRead = handler.EndReceive(ar);

// nếu còn có dữ liệu...

if (bytesRead > 0)

{

// chèn nó vào chuỗi chính

content += Encoding.ASCII.GetString(buffer, 0, bytesRead);

// nếu gặp ký tự đánh dấu hết chuỗi...

if (content.IndexOf(".") > -1)

{

Console.WriteLine("Read {0} bytes from socket. \n Data: {1}"/

content.Length, content);

byte[] byteData = Encoding.ASCII.GetBytes(content);

// gửi dữ liệu trở lại cho client

handler.BeginSend(byteData, 0 , byteData.Length, 0 ,

new AsyncCallback(SendCallback), handler);

}

else // tiếp tục nhận dữ liệu

{

handler.BeginReceive(buffer, 0 , buffer.Length, 0,

new AsyncCallback(ReceiveCallback), handler);

}

}

}

public static void SendCallback(IAsyncResult ar)

{

Console.WriteLine("SendCallback Thread ID:" +

AppDomain.GetCurrentThreadId());

Socket handler = (Socket) ar.AsyncState;

// gửi dữ liệu trở lại cho client

int bytesSent = handler.EndSend(ar);

Console.WriteLine("Sent {0} bytes to Client.", bytesSent);

// các thủ tục đóng socket

handler.Shutdown(SocketShutdown.Both);

handler.Close();

// thiết lập sự kiện cho thread chính

socketEvent.Set();

}

}