**Asynchronous Socket Programming in C#: Part I**

<https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm>

# Asynchronous Socket Programming in C#: Part I

Posted by **[Jayan Nair](https://www.codeguru.com/member.php/Jayan+Nair/)** on **July 26th, 2004**

Top of Form

* 0.25

0.50

0.75

1.00

1.25

1.50

1.75

2.00

2.25

2.50

2.75

3.00

3.25

3.50

3.75

4.00

4.25

4.50

4.75

5.00

* Vote!

Bottom of Form

### WEBINAR: On-Demand

Application Security Testing: An Integral Part of DevOps

[**~~Watch~~→**](http://o1.qnsr.com/cgi/r?WT.qs_dlk=WxIH3grIZ7wAAGjetvAAAAAT;;n=203;c=1643117;s=9479;x=7936;f=201707241206590;u=j;z=TIMESTAMP;k=https%3A%2F%2Fassetform.codeguru.com%2Fcontroller%3Fasset%3D237422710%26srvid%3D286710%26vkey%3D26272410%26io%3D226078%26qset%3DCONTACTFORM_HQB%26formHQB%3Dy%26domain%3Dwww.codeguru.com)

## Objective

The objective of this article is to demonstrate a socket-based **client/server** application that will allow two-way[**asynchronous communication**](http://www.umuc.edu/virtualteaching/module1/async.html) between a server and multiple client applications. Because this example uses asynchronous methods, the server application does not use threads to communicate to multiple clients (although internally the **asynchronous communication** mechanism uses threads at the OS level).

## The Difference Between Synchronous and Asynchronous Communication in Network Programming

The key difference between [synchronous](http://www.webopedia.com/TERM/S/synchronous.html) and [asynchronous](http://www.webopedia.com/TERM/A/asynchronous.html) communication can be explained with an example.

**https://www.codeguru.com/images/pdfImage.jpg**

**Remote Monitoring and Management: The Key to Proactive, Efficient IT Service Delivery**

[**Download Now**](https://o1.qnsr.com/cgi/r?WT.qs_dlk=WxIH3grIZ7wAAGjetvAAAAAT;;n=203;c=1395297;s=9479;x=7936;f=201308291424140;u=j;z=TIMESTAMP;k=https://assetform.codeguru.com/controller?asset=234875510&srvid=95910&vkey=10476310&io=232577&BLUID=2018060119571061025673:010476310:001&qset=CONTACTFORM_HQB&formHQB=y&domain=www.codeguru.com)

Consider a server application that is listening on a specific port to get data from clients. In **synchronous**receiving, while the server is waiting to receive data from a client, if the stream is empty the main thread will block until the request for data is satisfied. Hence, the server cannot do anything else until it receives data from the client. If another client attempts to connect to the server at that time, the server cannot process that request because it is blocked on the first client. This behavior is not acceptable for a real-world application where we need to support multiple clients at the same time.

In **asynchronous** communication, while the server is listening or receiving data from a client, it can still process connection requests from other clients as well as receive data from those clients. When a server is receiving asynchronously, a separate thread (at the OS level) listens on the **socket** and will invoke a callback function (specified when the asynchronous listening was commenced) when a socket event occurs. This callback function in turn will respond and process that **socket** event. For example, if the remote program writes some data to the socket, a "read data event" (callback function you specify) is invoked; it knows how to read the data from the socket at that point.

Even though this could be achieved by running multiple threads, the [**C#**](http://www.webopedia.com/TERM/C/C_sharp.html) and [**.NET framework**](http://www.webopedia.com/TERM/D/dot_NET_Framework.html)s provide a rich set of functionalities to do asynchronous communications without introducing the complexity of threading.

## Socket Class

The **Socket** class ( **[System.Net.Sockets.Socket](http://msdn.microsoft.com/en-us/library/system.net.sockets.socket.aspx)** ) provides a set of synchronous and asynchronous methods for synchronous or asynchronous communication. As per the **.NET** naming convention, all the asynchronous method names are created by prefixing the words "Begin" or "End" to the name of the synchronous methods. The methods prefixed with "Begin" and "End" represent a pair of asynchronous methods corresponding to a single synchronous method, as shown in the following table.

|  |  |
| --- | --- |
| **Synchronous Methods** | **Asynchronous Methods** |
| 1. Connect() | 1. BeginConnect() 2. EndConnect() |
| 1. Receive() | 1. BeginReceive() 2. EndReceive() |

## Example Application

The example shown in this article has two classes, one implementing the **Socket Server** and the other implementing the **Socket Client**.

   
Figure 1

The Socket Server application is implemented in the SocketServer class (file name SocketServer.cs). This class has a main Socket object (m\_mainSocket) and an array of worker Socket objects (m\_workerSocket) as members. The main Socket object does the listening for the clients. Once a client is connected, the main Socket transfers the responsibility to process the transactions related to that particular client to a worker Socket. Then, the main Socket goes back and continues listening for other clients.

BeginAccept() and BeginReceive() are the two important methods in the Socket class used by the Socket Server application.

The BeginAccept() method has the following signature:

1. public IAsyncResult BeginAccept(
2. AsyncCallback callback, // (1) Function to call when a client
3. // is connected
4. object state // (2) State object to preserve socket
5. // info
6. );

Essentially, after calling the Listen() method of the main Socket object, you call this asynchronous method and specify a call back function (1), which you designated to do the further processing related to the client connection. The state object (2) can be null in this particular instance.

Because this is an asynchronous method, it will return immediately and the server main thread is free to process other events. Behind the scenes, a separate thread will start listening on that particular socket for client connections. When a client requests a connection, the callback function you specified will be invoked.

Inside the callback function (in the example, the function is named "OnClientConnect()"), you will do further processing related to the client connection.

1. public void OnClientConnect(IAsyncResult asyn)
2. {
3. try
4. {
5. // Here we complete/end the BeginAccept() asynchronous call
6. // by calling EndAccept() - which returns the reference to
7. // a new Socket object
8. m\_workerSocket[m\_clientCount] = m\_mainSocket.EndAccept (asyn);
9. // Let the worker Socket do the further processing for the
10. // just connected client
11. WaitForData(m\_workerSocket[m\_clientCount]);
12. // Now increment the client count
13. ++m\_clientCount;
14. // Display this client connection as a status message on the GUI
15. String str = String.Format("Client # {0} connected",
16. m\_clientCount);
17. textBoxMsg.Text = str;
19. // Since the main Socket is now free, it can go back and wait
20. // for other clients who are attempting to connect
21. m\_mainSocket.BeginAccept(new AsyncCallback
22. ( OnClientConnect ),null);
23. }
24. catch(ObjectDisposedException)
25. {
26. System.Diagnostics.Debugger.Log(0,"1","\n OnClientConnection:
27. Socket has been closed\n");
28. }
29. catch(SocketException se)
30. {
31. MessageBox.Show ( se.Message );
32. }
34. }

The first thing you do inside the "OnClientConnect()" function is to call the EndAccept() method on the m\_mainSocket member object, which will return a reference to another socket object. You set this object reference to one of the members of the array of Socket object references you have (m\_workerSocket) and also increment the client counter. Now, because you have a reference to a new socket object that now can do the further transaction with the client, the main Socket (m\_mainSocket) is free; hence, you will call its BeginAccept() method again to start waiting for connection requests from other clients.

On the worker socket, you use a similar strategy to receive the data from the client. In place of calling BeginAccept() and EndAccept(), here you call BeginReceive() and EndReceive(). This, in a nutshell, is the Socket Server implementation. While you are sending out data to the clients, the server simply uses the specific worker socket objects to send data to each client.

   
Figure 1

The Socket Client application is implemented in the SocketClient class (file name SocketClient.cs). Compared to the server where you have a main Socket and an array of worker Sockets, here you only have a single Socket object (m\_clientSocket).

The two important methods in Socket class used by the Socket Client application are the Connect() and BeginReceive() methods. Connect() is a synchronous method and is called to connect to a server that is listening for client connections. Because this call will succeed/fail immediately, depending on whether there is an active server listening or not at the specified IP and Port number, a synchronous method is okay for this purpose.

Once a connection is established, you call the BeginReceive() asynchronous function to wait for any socket write activity by the server. Here, if you call a synchronous method, the main thread on the client application will block and you will not be able to send any data to the server while the client is waiting for data from the server.

When there is any write activity on the socket from the server end, the internal thread started by BeginReceive() will invoke the callback function ("OnDataReceived()" in this case), which will take care of the further processing of the data written by the server.

When sending the data to the server, you just call the Send() method on the m\_clientSocket object, which will synchronously write the data to the socket.

That is all there is for asynchronous socket communication using multiple clients.

## Limitations/Possible Improvements

* Up to 10 simultaneous clients are supported. You can easily modify and support unlimited number of clients by using a HashTable instead of an array.
* For simplicity, when the server sends out a message, it is broadcast to all the connected clients. This could easily be modified to send messages to specific clients by using the Socket object pertaining to that particular client.
* When a client is disconnected, proper action is not taken; the client count is not decremented. The correct way would be to reuse or release resources for other client connections.

## Acknowledgement

Even though the content of this article is independently developed, the example program used is influenced by the article on [Socket Programming in C# by Ashish Dhar](http://www.developerfusion.co.uk/show/3997/).

##### Update added on 03/01/2005

**For a more comprehensive example covering topics such as thread synchronization, please see**[**Part II**](http://www.codeguru.com/Csharp/Csharp/cs_network/sockets/article.php/c8781/)**of this article.**

## About the Author

#### Jayan Nair

**Jayan Nair** is a Senior Software Engineer with 15+ years of experience working with cutting edge software technologies. Currently he is developing the next generation software applications for the telecommnunications testing industry. Jayan's passions: Object Oriented software design and developing reusable software components. His motto: "if the software you write is not reusable, you are not writing **soft**ware, but **hard**ware instead". Jayan finished his Masters degree in Computer Science from [Virginia Tech](http://www.vt.edu/), Blacksburg, VA. His expertise includes, C, C++, Java, J2EE, Visual Basic, C#, ASP.NET and distributed applications. He is also a Sun Certified Programmer for the Java Platform (SCPJ). You can contact him at jnair1998@hotmail.com.

## Related Articles

* [Asynchronous Socket Programming in C#: Part II](https://www.codeguru.com/csharp/csharp/cs_network/sockets/article.php/c8781/Asynchronous-Socket-Programming-in-C-Part-II.htm)
* [Communicating over Sockets: Blocking vs. Unblocking](https://www.codeguru.com/csharp/csharp/cs_network/sockets/article.php/c11649/Communicating-over-Sockets-Blocking-vs-Unblocking.htm)

## Downloads

* [async\_client\_server\_exe.zip](http://www.codeguru.com/dbfiles/get_file/async_client_server_exe.zip?id=7695&lbl=ASYNC_CLIENT_SERVER_EXE_ZIP)
* [async\_client\_server\_src.zip](http://www.codeguru.com/dbfiles/get_file/async_client_server_src.zip?id=7695&lbl=ASYNC_CLIENT_SERVER_SRC_ZIP)

## Comments

#### software architecture

Posted by asad on 08/29/2016 08:42am

This blog awesome and i learn a lot about programming from here.The best thing about this blog is that you doing from beginning to experts level. Love from Pprogramming

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#58471022)

#### Project with windows forms

Posted by Gabriel da Cruz Moreira Falieri on 06/22/2014 08:29am

if possible send me this project with all forms that have been created ... I saw that works perfectly

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#2028057323)

#### Thank you very very much.

Posted by Jadav Vishal on 06/12/2014 07:29am

I just want to say thank you very much for the code, you saved my ass. thank you again.

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#1976057323)

#### Picture

Posted by Darc on 03/17/2013 11:18pm

Excuse me, but some picture is gone can it be re uploaded or anything???

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#11183322)

#### source code is not working

Posted by Shefeek on 07/04/2012 03:23am

Hi, When i used the exe it is working fine,but when i use your source code it gives some exception like invalid operation exception.Please reply to me. Regards Shefeek

#### I've got the same problem

Posted by Mit on 11/05/2012 10:27pm

When I copied the source in a new empty project, the visual studio can not run it and brings me back with the same problem. However, running the exe file created by that project,l works well!! what's the problem?

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#4278222)

#### I have the same problem

Posted by Mitt on 11/05/2012 10:24pm

Even, I copied the code in an empty project and ran it, again I got the same problem. However, when I try to use the exe file created by this new project, it works well!!!! what's the problem when executing with visual studio?

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#131201523)

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#3771323)

#### thanks for you

Posted by smaas on 05/29/2012 12:57pm

thankyou very

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#2781023)

#### Errors

Posted by mlam56 on 12/16/2011 05:14am

When I use your Exe, the system works great. However when I imp[lant the cs file into my C# 2010 Express envirement it will not run. I get an comment on GetHiostByName, should be replaced bij GetHostEntry. It works but is not giving me an IPv4 address. Then I get a error on the IPStr = ipaddress.ToString(); The system stops.

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#509360)

#### Cross-thread operation not valid

Posted by LMAI on 07/15/2010 01:18pm

I love this example - for newbie like me. When I use both exe files for Client PC and Server PC, every is o.k. Then I use SockeClient.exe file to run on the Client PC and source code SocketServer.cs on other PC as a client. The Server sending a msg to Client is o.k But, when Client sends msg to Server, the Cross-thread Operation not valid message takes place. at line richTextBoxReceivedMsg.AppendText(szData); in method public void OnDataReceived(IAsyncResult asyn). Moreover, Client sends 123456 Server just get "1\0" on the szData Could anyone show me how to solve this problem? I use VS2008 Cheers.

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#262090)

#### Writing received message to screen.

Posted by Gytax on 10/21/2009 08:00am

Hello there,

I am quite new to Socket Programming and I'm creating a program using the example code explained here.

Now, when I receive a message from a client, I want to show:

Client 1: message.

I can't seem to get that to work: what I get now is:

Client 1: m Client 1: e...etc

public void OnDataReceived(IAsyncResult asyn)

{

try

{

SocketPacket socketData = (SocketPacket)asyn.AsyncState;

int iRx = 0;

iRx = socketData.m\_currentSocket.EndReceive(asyn);

String remEP = socketData.m\_currentSocket.RemoteEndPoint.ToString();

char[] chars = new char[iRx + 1];

System.Text.Decoder d = System.Text.Encoding.UTF8.GetDecoder();

int charLen = d.GetChars(socketData.dataBuffer, 0, iRx, chars, 0);

System.String szData = new System.String(chars);

txtLog.AppendText(szData + "\r\n");

WaitForData(socketData.m\_currentSocket);

}

catch(SocketException se)

{

MessageBox.Show(se.Message);

}

}

That's the code I am modifying.

Any help would be greatly appreciated.

#### Re: Writing received message to screen

Posted by Hemicharger99 on 11/03/2009 03:32pm

The problem is that the socketData.dataBuffer is only holding one character at a time, and when you are appending the txtLog textbox, it will display as

Client 1: m

Client 1: e

Client 1: s...etc

You can increase the size of the buffer to accomodate the size of the socket's "ReceiveBufferSize".

Also, if you are writing to the screen, be sure to use a delegate to update your textbox. I apologize, this is in VB.Net instead of c#, but you can understand the idea of it. i.e.

Delegate Sub DoUpdateTextBox(ByVal pMsg As String)

Public Sub UpdateTextBox(ByVal pMsg As String)

If (Me.InvokeRequired) Then

Me.BeginInvoke(New DoUpdateTextBox(AddressOf UpdateTextBox), New Object() {pMsg})

Else

txtLog.AppendText(pMsg + "\r\n")

End If

End Sub

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#257410)

[Reply](https://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695/Asynchronous-Socket-Programming-in-C-Part-I.htm#46894810)

#### Bug in the server code

Posted by Nilsen on 10/01/2009 04:28am

In the OnClientConnect-method on the server you must NOT call the BeginAccept after each connection! The main socket is already listening for more connections - doing this will result in memory being eaten! The number of simultaneous connections you specify when calling m\_MainSocket.Listen(x) is actually telling the socket class how many instances of OnClientConnect that can run SIMULTAENOUSLY. And even if you'd called .Listen(1), your code would still work perfectly since you're branching the data retrieval to a worker socket.

I had your code in a high-stress socket application recently and I couldn't for the love of my life find out why my program always seemed to crash after about 5-6 hours of runtime. Task Manager showed that it had grown from about 15 MB of memory usage to about 250. Turns out it finally reached a point where the Socket system of .NET got clogged up, and this exception was the result each time it called .EndAccept:

*An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full*

What put me on the right track was when I coded in a "Thread.Sleep(5000)" in the exception-catch. Even after I did this, the exceptions occured many times per second (I could see that from log files) so this indicated to me that there was MANY threads (hundreds) that were waiting for connections, even though I had called Listen(3).

After I removed the BeginAccept-line, my program can now run "forever" :-)

#### It's not a bug

Posted by Ryan on 11/10/2015 05:20pm

If the OnClientConnect method does not call BeginAccept, no more clients will be able to connect. I just confirmed this on an application that I am working on.