

Re: asynchronous socket communication

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-01/msg04396.html>

- *From:* "Ignacio Machin \(.NET/ C# MVP \)" <machin TA laceupsolutions.com>
 - *Date:* Thu, 25 Jan 2007 14:07:16 -0500
-

Hi,

Well that's a fair amount of code let me tell you. Did you try a sync connection first?

"panko" <PKoperski@xxxxxxxx> wrote in message
<news:1169747503.338678.124410@xx>

| Hello,
| I can't manage with asynchronous socket communication. :(
| I wrote a class CSocket.cs.
| This class is taking care of sending strings to LED display. This
| display is actually communicating via serial port and serial/ethernet
| converter (MOXA NE-4100T) with TCP server. So communication is in that
| way:
|
| MyApplication(TCP
| client)------(TCPserver)MOXA(serial)---(serial)LED Display
|
| In the constructor there are three parameters:
| public CSocket(IPAddress p_ipAddress, int p_port, ArrayList
| p_arrayToSend)
|
| The array is array of strings that are valid for special LED display
| protocol. The server is giving reply OK or BAD if the string it
| received is display
|
| protocol valid (syntax, crc etc).
| The main public method of that class is
| public void ConnectAndSend()
| {
| if (this.m_status != EStatus.Connected)
| {
| res = this.SequenceConnect();
| }
|
| this.m_iTriedSend = 0;

Re: asynchronous socket communication

```
| if (this.m_status = EStatus.Connected)
| {
| while(counter<p_arrayToSend.Length || this.m_iTriedSend <
| REPEATSEND)
| {
| Send(CurrStrFromArray);
| this.m_iTriedSend++;
| if (this.m_status==EStatus.Received) counter++;
| }
| }
| }
| This is only abstract of that method to give general idea.
| The SequenceConnect tries to connect using Socket.BeginConnect methd.
| When connection is established OnClientConnect method is called to call
| Socket.EndConnect method.
| This process I wanted to have in the ConnThread . When the thread is
| finished (connection established or not) the main thread can go further
| to send data (or not)
|
| The same idea idea is with sending the strings to TCP server. When the
| data is sent ReceiveThread is launched to wait for the reply and when
| there is response from the server, the main thread will keep on sending
| following strings.
|
| Here is most important part of the code. I hope names are clear enough
|
| private void SequenceConnect()
| {
| this.m_iTriedConnect = 0;
| while (this.m_iTriedConnect < REPEATCONNECT)
| {
| ThreadStart privThreadDeleg = new ThreadStart(TryConnect);
| Thread ConnThread = new Thread(privThreadDeleg);
| ConnThread.Start();
| // wait until thread finish or time elapses
| ConnThread.Join(TIME_ELAPSED);
| if (this.m_status == EStatus.Connected) break;
| }
| }
|
| public void TryConnect()
| {
| this.m_iTriedConnect++;
| try
| {
| IPEndPoint remoteEndPoint = new
| System.Net.IPEndPoint(this.m_ipAddress, this.m_iPort);
| this.m_socketClient.BeginConnect(remoteEndPoint,new AsyncCallback (
| OnClientConnect ),null);
| this.m_status = EStatus.Connecting;
| }
| }
```

Re: asynchronous socket communication

```
| catch (SocketException e)
| {
| else this.m_status = EStatus.ConnError;
| }
| return;
| }
|
| private void OnClientConnect(IAsyncResult asyn)
| {
| try
| {
| this.m_socketClient.EndConnect(asyn);
| if(this.m_socketClient.Connected) this.m_status = EStatus.Connected;
| else this.m_status = EStatus.ConnError;
| }
| catch (SocketException e)
| {
| else this.m_status = EStatus.ConnError;
| }
| return;
| }
|
| private void Send(string p_stringToSend)
| {
| this.m_status=EStatus.Sending;
| try
| {
| Object objData = p_stringToSend;
| byte[] byData = Encoding.ASCII.GetBytes(objData.ToString ());
| this.m_socketClient.Send (byData);
| this.m_status = EStatus.Sent;
|
| ThreadStart privThreadDeleg = new ThreadStart(WaitForData);
| Thread ReceiveThread = new Thread(privThreadDeleg);
| ReceiveThread.Start();
| // wait until thread finish or time elapses
| ReceiveThread.Join(TIME_ELAPSED);
| }
| catch (SocketException e)
| {
| this.m_status = EStatus.SendError;
| }
| return;
| }
|
| private void WaitForData()
| {
| this.m_status = EStatus.Receiving;
| try
| {
| IAsyncResult m_asynResult =
```

Re: asynchronous socket communication

```
|  
| m_socketClient.BeginReceive(m_bDataBuffer,0,8,SocketFlags.None,pfnCallBack,null);  
| }  
| catch (SocketException e)  
| {  
| this.m_status = EStatus.ReceiveError;  
| }  
| return;  
| }  
|  
| private void OnDataReceived(IAsyncResult asyn)  
| {  
| try  
| {  
| int iRx = 0;  
| iRx = m_socketClient.EndReceive(asyn);  
| char[] chars = new char[iRx + 1];  
| Decoder d = Encoding.UTF8.GetDecoder();  
| this.m_iCharLen = d.GetChars(m_bDataBuffer, 0, iRx, chars, 0);  
| this.m_Received = new String(chars);  
| this.m_status = EStatus.Received;  
| }  
| catch (SocketException e)  
| {  
| this.m_status = EStatus.ReceiveError;  
| }  
| return;  
| }  
|
```

| The problem I found here is that application doesn't stop until
| ConnThread is finished. Main thread passes
| "ConnThread.Join(TIME_ELAPSED);" and goes to next statement which is
| "if (this.m_status == EStatus.Connected) break;" while the value is
| Connecting – so next loop is done. How can I prevent it?
| Sometimes main thread is waiting for "ConnThread.Join(TIME_ELAPSED);" statement.
| I don't understand why. Does AsyncCallback start a new thread?
| I think I have some problem with threading here.
| Or maybe you know some better way/idea to solve that kind of
| communication.

| Regards,
| panko
|