

Re: Socket Exception

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.networks/2006-10/msg00134>

- *From:* Dave <Dave@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 6 Oct 2006 10:14:02 -0700
-

Alexander,

Thanks for your comments. My test app is only to test some code that is behind some web services pages. You offered some design changes that I would be interested in making. However, how would your input change knowing that this code resides in an .asmx page to support web service calls?

My goal is to take the web service information and get it to my .exe process to do the actual work.

"Alexander Nickolov" wrote:

When the socket is writable, that doesn't mean you have a connection. You may have an error to be reported on the first write. In this case the server most likely rejected your connection (sent back a RST). I'm not sure if localhost is subject to the same queue limits as regular network interfaces, but if it is, and you are running on a client OS (e.g. not a server OS like Win2k3 Server), you may be exceeding the TCP backlog limit of 5 (server OSes have backlog of 200) unACKed SYNs.

Overall I'd question your design of opening a new connection for each command. Why not use a pool of connections and pick the first available for a roundtrip exchange? That's just one of the available alternatives obviously – the simplest to retrofit into your existing code...

--

=====
Alexander Nickolov
Microsoft MVP [VC], MCSD
email: agnickolov@xxxxxxx
MVP VC FAQ: <http://www.mvps.org/vcfaq>
=====

"Dave" <Dave@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message

Re: Socket Exception

news:8B0A50DB-B04E-4283-BF10-5AAC08D5DCF9@xxxxxxxxxxxxxxxxxxxx

I have written a small test app that causes this problem in about one or two seconds. Both the client (test app below) and the server run on the same machine. The test app starts 10 threads that send three commands to the server. Each command is a new socket connection that is opened and closed at each command send. I am not sure if this is a problem with the way I am using the .NET Socket libraries or the settings I have chosen for the connection.

On the server side no socket message is received when the exception occurs.

The server never sees anything about that connection.

Client Test Code.....

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace SocketExceptTest
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }
    }
}
```

Re: Socket Exception

```
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support – do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(64, 80);
    this.button1.Name = "button1";
    this.button1.TabIndex = 0;
    this.button1.Text = "Start...";
    this.button1.Click += new System.EventHandler(this.button1_Click);
    //
    // Form1
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(292, 273);
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);

}
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
```

Re: Socket Exception

```
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void Form1_Load(object sender, System.EventArgs e)
{
}

private string SendNetCommand(string outCommand)
{
    string retString = "Error sending command.\n";

    // Setup socket connection to the video server
    TcpClient tcpClient = new TcpClient();
    try
    {
        LingerOption lingerOption = new LingerOption(false, 0);
        tcpClient.LingerState = lingerOption;
        tcpClient.NoDelay = true;
        tcpClient.Connect("localhost", 2345);
        NetworkStream networkStream = tcpClient.GetStream();
        if(networkStream.CanWrite && networkStream.CanRead)
        {
            // Does a simple write.
            Byte[] sendBytes =
            Encoding.UTF8.GetBytes(outCommand);//.ASCII.GetBytes(outCommand);
            networkStream.Write(sendBytes, 0, sendBytes.Length);

            // Reads the NetworkStream into a byte buffer.
            byte[] bytes = new byte[tcpClient.ReceiveBufferSize];
            int readLen = networkStream.Read(bytes, 0, (int)
            tcpClient.ReceiveBufferSize);

            // Returns the data received from the host to the console.
            char[] trimChars = {'\n', '\r', ' '};
            retString = Encoding.UTF8.GetString(bytes).Substring(0,
            readLen).TrimEnd(trimChars);
        }
        else if (!networkStream.CanRead)
        {
            retString = "Error – socket to language server is not allowing reads.";
        }
        else if (!networkStream.CanWrite)
        {
            retString = "Error – socket to language server is not allowing writes";
        }
        networkStream.Close();
        tcpClient.Close();
    }
}
```

Re: Socket Exception

```
catch (Exception e )
{
retString = "Error – socket exception, " + e.ToString();
tcpClient.Close();
}
return retString;
}

private void button1_Click(object sender, System.EventArgs e)
{
int i = 0;

// create some threads for testing
for(i = 0; i < 10; i++)
{
Thread thread = new Thread(new ThreadStart( CommandThread ));
thread.Name = "dt" + i.ToString();
thread.Start();
}
}

protected void CommandThread()
{
String name = System.Threading.Thread.CurrentThread.Name;

// Create threads and endlessly loop on the sending of commands to the
video server
int k = 0;
for(k = 0; k < 1000; k++)
{
SendNetCommand("create command\n");
SendNetCommand("do someting\n");
SendNetCommand("destroy command\n");
}
}
}
}
```

"Dave" wrote:

I have a .NET program that makes a socket connection to another process on the same machine. A socket connection is opened, then data sent, then socket is closed. This happens several times a second. Most of the time the

Re: Socket Exception

calls
are successful. However, sometimes I get the following
exception:

Error – socket exception, System.IO.IOException: Unable to
write data to
the
transport connection. --->
System.Net.Sockets.SocketException: An
existing
connection was forcibly closed by the remote host

When I look at the destination application log I don't see any
new socket
open when this happens. The next call to the socket will
succeed.

I am not sure what to look for. I have tried retrying the
socket
connection
but that does not work.

What is that exception really telling me? What is the correct
interpretation
of the exception?

Thanks,
Dave