

Re: Console app freezes

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.compactframework/2004-12/1259.htm>

From: Ilya Tumanov [MS] (ilyatum_at_online.microsoft.com)

Date: 12/21/04

Date: Tue, 21 Dec 2004 01:43:05 GMT

What's in WaitForCommEvent()? It's not a while loop which checks for some variable set from event, is it?

I would also suggest using circular buffer instead of creating new array lists and arrays for every packet.

You'll need to figure out what to do in case buffer overflows (terminate, log event, ignore extra data, etc.).

Best regards,

Ilya

This posting is provided "AS IS" with no warranties, and confers no rights.

> *Thread-Topic:* Console app freezes
> *thread-index:* AcTm7jZvbF/weKNQT22YrExp4stGvg==
> *X-WBNR-Posting-Host:* 202.6.138.45
> *From:* "?Utf-8?B?TWLjaGFlbC0tSg==?" <MichaelJ@discussions.microsoft.com>
> *References:* <2A88D523-2302-4891-943E-2170F347100A@microsoft.com>
<41c11d2a\$1@news.microsoft.com>
<BE00D6F5-A077-47D8-9C4A-3528F9494CE9@microsoft.com>
<KNzYqk64EHA.3512@cpmsftngxa10.phx.gbl>
<728DC81F-04A6-43A7-BCE5-D4132D86CDF6@microsoft.com>
<UQI8R\$r5EHA.2600@cpmsftngxa10.phx.gbl>
> *Subject:* Re: Console app freezes
> *Date:* Mon, 20 Dec 2004 15:47:06 -0800
> *Lines:* 116
> *Message-ID:* <17826F3A-DA24-4C76-A645-7DF5087B4805@microsoft.com>
> *MIME-Version:* 1.0
> *Content-Type:* text/plain;
> *charset="Utf-8"*
> *Content-Transfer-Encoding:* 8bit
> *X-Newsreader:* Microsoft CDO for Windows 2000
> *Content-Class:* urn:content-classes:message
> *Importance:* normal
> *Priority:* normal
> *X-MimeOLE:* Produced By Microsoft MimeOLE V6.00.3790.0

> *Newsgroups: microsoft.public.dotnet.framework.compactframework*
> *NNTP-Posting-Host: TK2MSFTNGXA03.phx.gbl 10.40.1.29*
> *Path: cpmsftngxa10.phx.gbl!TK2MSFTNGXA03.phx.gbl*
> *Xref: cpmsftngxa10.phx.gbl*
microsoft.public.dotnet.framework.compactframework:67289
> *X-Tomcat-NG: microsoft.public.dotnet.framework.compactframework*
>
> *My app uses OpenNETCF.org's serial comms code with a few modifications*
in
> *their Port.cs class. This class simply sets up a thread called*
> *CommEventThread that continually loops the associated com port for*
data.
> *However, I had to modify this to reflect the modifications I made to the*
> *lower level serial drivers. The serial drivers were changed such that it*
> *suited the data it was processing. The data was basically composed of*
packets
> *separated by parity errors. I couldn't use a parity error event to*
> *distinguish between packets because the app couldn't keep up with the*
number
> *of parity error events being raised. Instead, I modified the serial*
driver
> *such that it places an escape character in the receive stream*
whenever
> *it detected a parity error and my app simply needed to locate this*
character.
> *To distinguish between an escape character between an*
actual
> *data character, I replaced the data one with a 6-byte signature which my*
higher
> *level app also searches for. So in pseudo code, this is how the*
> *CommEventThread in Port.cs looks like:*
>
> *private void CommEventThread()*
> *{*
> *while(hPort != invalidHandle)*
> *{*
>
> *// Wait for a comm event to take place;*
> *WaitForCommEvent();*
>
> *if(error event)*
> *{*
> *// Handle error;*
> *}*
>
> *if(data received)*
> *{*
> *do*
> *{*
> *ReadFile() to rxFIFO buffer;*
> *}while(bytesread > 0)*

```
> }
>
> while(rxFIFO.Count > 0)
> {
> byte u = rxFIFO.Dequeue();
>
> if(u == any byte)
> {
> // Add ~u™ to progressBuffer ArrayList
> progressBuffer.Add(u);
> }
>
> if(u == ~f0™)
> {
> // Raise ParityError() event and clear progressBuffer;
> ParityError();
> progressBuffer = new ArrayList();
> }
>
> if(u is part of 6–byte signature)
> {
> // Take note of it;
> }
>
> if(u is last byte of 6–byte signature)
> {
> // Insert ~f0™ in the data;
> progressBuffer.Add(u);
> }
> }
> }
> }
>
> I also added the method ReadProgressBuffer() in Port.cs which gets the
> current progressBuffer contents when a parity error event is raised. This
> will be called by my main app.
>
> public byte[] ReadProgressBuffer(){
> return (byte[])progressBuffer.ToArray(typeof(byte));
> }
>
> In my main code, I instantiate 4 port objects which means that I
introduce 4
> CommEventThreads executing the above code in non–stop looping. My main
also
> handles the ParityError events raised by each port by creating 4
identical
> event handlers that process the data by firstly calling
> portX.ReadProgressBuffer(). Below is the pseudo code:
>
> private void portX_ParityError()
```

```
> {  
> byte[] packet = portX.ReadProgressBuffer();  
>  
> // Process the byte array here;  
> // This predominantly involves packet validation –  
> // CRC checks, and updating a set of global data;  
> }  
>  
> When this handler returns, the CommEventThread continues by clearing the  
> progressBuffer and repeating the whole process.  
>  
> In addition to this, I have 3 timers, TCP/IP stuff, and log keeping (as  
> described in my first post). Would it definitely be a CPU overload issue?  
If  
> you need more information regarding my app or any of its code, please  
feel  
> free to ask. Thanks again.  
>  
>  
> ""Ilya Tumanov [MS]"" wrote:  
>  
> > This behavior seem to be a result if excessive CPU load.  
> > Do you have some tight loops anywhere? Say, waiting for something?  
> > Do you know what your app will do in case more data is coming from  
serial  
> > port(s) than can be processed?  
> > Could it wait for available buffer in a loop without sleep() in it?  
> > That would cause the "serial" thread to use 100% CPU, which would slow  
down  
> > all other thread, which would slow down data processing, which would  
cause  
> > "serial" thread to wait... and so on. As you remove incoming data, your  
app  
> > would eventually process data and unlock itself.  
> >  
> > Best regards,  
> >  
> > Ilya  
> >  
> >  
>
```