

Re: How do you kill a completely locked up thread?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-01/msg02074.html>

- *From:* TheSilverHammer <TheSilverHammer@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 16 Jan 2008 12:43:03 -0800
-

"Jeroen Mostert" wrote:

I'm not an MSVP but I've seen this so many times in our codebase that it's not funny anymore. The one way to cancel pending I/O on a socket and unwedge threads blocking on that is to close the socket from another thread and handle the resulting exceptions. Nothing else will do, at least nothing that can be called reliable. Of course, this means tearing down the connection, but that's still a whole lot better than tearing down your process.

The other alternative, which is less straightforward but suits some designs better, is to make sure that threads never issue I/O which can take "forever". Almost every I/O call has a timeout parameter, and for those that don't there's always asynchronous I/O and `ThreadPool.RegisterWaitForSingleObject()`. When the call returns with a timeout, either poll, decide to wait some more or give up and close the socket, which you can then do from the same thread that owns the socket, simplifying error handling.

I understand it's not your code, but trust me: you'll want to rewrite it anyway, unless you can afford restarting your application every so often.

Grrr.. Damm post thing asked me to login again and ate me post... Anyway...

The SharpSSH code base has a bunch of classes and would take a major effort to re-write. It is clear that it is unfinished from looking at it. I am not sure my company wants to fund me re-writing this code set.

However, the solution Peter Bromberg gave on his web site looks good except for what appears to me to be a big hole or leak. I do not understand how C# handles this, so maybe it is a non issue. The following is the code segment from his web site, I hope he doesn't mind me posting it:

```
public ArrayList DoWorkNeedsTimeout(ArrayList alin, int secondsToWait)
{
    ArrayList alOut = new ArrayList();
```

Re: How do you kill a completely locked up thread?

```
//Create an instance of our delegate, pointing to the helper
method:

DoWorkNeedsTimeoutDelegate deleg = new
DoWorkNeedsTimeoutDelegate(DoWorkWithTimeout);

// Call BeginInvoke on delegate.
// Note on last two parameters of Delegate BeginInvoke Method:
// 1) callback: not used here, we can pass null
// 2) state: not used, pass an instance of object in the required
parameter location
// Invoke the delegate passing the parameters and get the
IAsyncResult object in "ar":

IAsyncResult ar = deleg.BeginInvoke(alin, secondsToWait, null,
new object());

// if the WaitOne method times out before we get a result, it
will be false:
if (!ar.AsyncWaitHandle.WaitOne(5000, false))
{

// handle timeout logging / notification here – Syslog,
Database, Email – whatever you need
alOut.Add("TIMED OUT!");
}

else // we didn't time out:
{
// get the result of the method call here
alOut = deleg.EndInvoke(ar);
}

return alOut;

}
```

What he is doing is making a delegate to call `BeginInvoke` with and then using the `IAsyncResult` to wait for a time period. If the time period expires, then his thread continues on. If it doesn't expire, he calls `EndInvoke()`. This looks good except for the issue of dealing with a truly locked-up thread.

`BeginInvoke()` uses a thread from the thread-pool right? So what happens if that thread never returns so you can call `EndInvoke`? Is it gone from the thread-pool forever? If you repeat this look 1000s of times and even if 1% of the time you get a locked up thread, won't you run out of threads?

The only way this can work indefinitely, which it may, is if the Garbage collector will reclaim the thread once the delegate and other related objects

Re: How do you kill a completely locked up thread?

Re: How do you kill a completely locked up thread?

are out of scope. Is this how it works?