

Re: Can NetworkStream.Write() block?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-01/msg02670.html>

- *From:* Jeroen Mostert <jmostert@xxxxxxxxxx>
 - *Date:* Sat, 19 Jan 2008 21:15:49 +0100
-

Gaz wrote:

```
myNetworkStream.Write(byte[], int offset, int size);
```

<snip>

It seems this is better than using the non-blocking version
`myNetworkStream.BeginWrite(byte[] buffer, int offset, int size)` since all that does (if the write buffer is full) is create a backlog of `BeginWrite` threads all waiting to complete.

This is **not** what happens. The `BeginX` methods generally do **not** create threads at all to handle the asynchronous requests, because that doesn't scale. Instead, they use overlapped I/O when possible. Do not assume that the asynchronous methods are built on the synchronous methods, because it's generally the other way around.

In general, you can assume asynchronous calls are **more** efficient than their synchronous companions, since those **always** take up a thread (namely the thread waiting for completion) while the asynchronous ones generally only take up a thread when executing the completion callback.

Might as well use the blocking version and increase the buffer size if that is possible.

For overlapped I/O (which .NET uses whenever possible) there's no blocking. The overlapped requests are simply queued until there's an opportunity to process them. In the rare event that the requests (plus their associated buffers) overflow the queue, you'd get a `SocketException`, but I'm almost certain there's no blocking. (Almost certain -- if a .NET networking guru could pipe up and confirm or deny this, I'd be grateful.)

—

J.

.