

## Re: Unicast UDP Server

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-03/msg03793.html>

---

- *From:* "Peter Duniho" <[NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx](mailto:NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sat, 29 Mar 2008 15:58:52 -0700
- 

On Sat, 29 Mar 2008 15:18:19 -0700, O.B. <[funkjunk@xxxxxxxxxxxxxxxx](mailto:funkjunk@xxxxxxxxxxxxxxxx)> wrote:

Wow, this is great. All this time I have been confusing bind and connect. Thank you for the detailed explanation.

You're welcome.

For the record, I still have to bind to a `_local_` IP address for UDP sockets in order for the receive data callback to be invoked. From what I gathered from your post, invoking "connect" with a remote address in a UDP socket will ensure that the receive callback is only invoked with data received from the specified remote address, correct?

It should, yes.

[...]

The code has been rewritten using unmanaged code to get a significant speed up over the original C++ implementation. Without the 75 MB receive buffer, we `_occasionally_` find the buffer overflowing.

I don't understand the above comment. If you're using unmanaged code, why are you posting in the C# newsgroup? C# is dependent on the .NET runtime, and so is inherently tied to managed code. If the original implementation was C++, is the above statement meant to imply that you were using managed C++?

In running a profiler, it appears the code is processing the data as fast as the OS is invoking the asynchronous callback. As a test, I wrote a receive callback that only sucked data off the socket and put it into an internal buffer for processing. We were still encountering an overflow ... very odd.

How do you know you are encountering an overflow? Are you sure that you aren't losing datagrams due to some other reason?

## Re: Unicast UDP Server

Maybe it is the actual NIC we're using? Or is it a limitation of Windows XP and Windows 2003 Server? This is a tough one to figure out.

The "U" in "UDP" might as well stand for "unreliable". UDP is inherently unreliable, and you will lose datagrams eventually, no matter what you do.

On a LAN, you can avoid many of the problems that exist going over the Internet, but your code still must always be able to deal with lost datagrams. Data loss will be reduced significantly on a LAN as compared to the Internet, but it will never go away completely.

You haven't made any mention as to what your actual network speed and load is, but I'll reiterate my previous statement: only on the fastest networks, under the highest loads, should a buffer of 75MB be useful. Even at that size, that's on the order of 500 ms worth of constant data throughput on a 1 gigabit network. Or put another way, if a 75MB buffer is needed, that means that the layer of network transport that's supposed to be reading from that buffer is at times being delayed by longer than 500 ms.

On a modern computer, that's practically forever. In my opinion, you should never be seeing delays that long. If you are seeing delays that long, there's probably a better way to deal with them. Such as, preventing the delays rather than allowing them to happen and just letting the data pile up in a larger buffer.

As an example of that 500 ms really means: you'd have to be running roughly ten fully CPU-bound threads on a single-core/CPU computer in order for some other thread's execution to be delayed by that much time. And of course, running ten fully CPU-bound threads on a single-core computer is definitely not a good idea (heck, running just two fully CPU-bound threads on a single-core computer isn't a good idea).

If you're running on a more conventional network (e.g. 100 Mb/s...though I admit, 1 Gb/s is getting more and more common, I think 100 Mb/s is still more typical), and/or your data transmission rate does not saturate the network, then a 75MB buffer implies an even larger delay (for a 100 Mb/s, it's more like 5 seconds than 500 ms). And if a half-second delay is a sign of something wrong, you can imagine that an even larger delay means. :)

Pete

.