

Re: Proxying bytes from one socket to another...

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.performance/2004-02/0043.html>

From: Tom Hall (*nospam_at_harmonyit.com*)

Date: 02/26/04

Date: Wed, 25 Feb 2004 21:50:33 -0500

200ms sounds suspiciously familiar – check into the Nagle algorithm which waits up to 200ms attempting to prevent excess network traffic by waiting for additional data to be gathered into one packet (send coalescing) – note this only affects small packets like you said. You can turn it off – its a socket option (SocketOption.NoDelay). You may need to turn it off on server and client.

Also, if you do a network trace you may see delayed acknowledgement (also 200ms) from the client end where the socket is waiting after receiving several packets hoping for you to send some data back that it can piggyback the ACK packets on top of. After about 200ms it gives up and sends the ACK(s) by themselves – then the server is free to send more data. And it repeats. This is normal socket (TCP) protocol.

Typically you will see this if the server is sending only and the client end is receiving only. That is the client doesn't have any sort of reply to send back after each message. The sending of a reply in this case avoids this situation and will cause a tremendous increase in speed but also causes unnecessary traffic. I did this in a protocol I developed for my application and response went from 200ms to <1ms between packets. You should only do this if you have lots of small (<MTU) packets to send and you can't coalesce them together or you NEED (my case) to get the data there now, not 200ms from now.

If you haven't done a network sniff using Ethereal www.ethereal.com you should (its free) – its invaluable to really see what's going on.

You can avoid the array copy if you **know** how big the header is before hand (sounds like you do – you said 2 ints). Leave room at the beginning of the byte[] for the header, then use the Socket.Receive call that looks like this:

```
int Receive(byte[] buffer, int offset, int size, SocketFlags socketFlags);
```

In your case, do

```
int recv=conn.Receive(buff,8,1024,SocketFlags.None);
```

microsoft.public.dotnet.framework.performance: Re: Proxying bytes from one socket to another...

Now, your data is in the array starting at byte 8, leaving room for you to stick the header in and hand the whole thing off to the MessagePacket. Of course MessagePacket will have to be passed and track the length including header as you are handing in the whole 1024 byte buffer – some of which won't have data in it.

Are you sure you don't need a bigger buffer, the MTU on Ethernet is 1500 bytes so if you ever get packets bigger than your 1024 it will end up split up as one big and one small receive typically which you would then forward as two sends. That's what the Nagle algorithm tries to avoid in the first place!

HTH
Tom

"Kevin Z Grey" <anonymous@discussions.microsoft.com> wrote in message news:E84BACE9-2F7C-4EF7-B36B-0B47081880EB@microsoft.com...

> *Heya,*

>

> *I have a proxy server that essentially encapsulates received data by pre-pending a few integer values for each chunk of data read off the socket. Now when the data is read, it is constructed into a "Message" object which is basically a struct of int, int, byte[], and is then dumped into a queue to schedule it to be sent by another Thread. Since I have multiple Threads, I am forced to make a copy of the incoming byte array and queue the copy so that my data does not get fragmented. The resulting code looks like this:*

>

> *byte[] buff = new byte[1024];*

> ...

> *if (conn.Available > 0)*

> {

> *// Read Bytes*

> *int recv = conn.Receive(buff);*

> *byte[] t_buff = new byte[recv];*

> *Array.Copy(buff, 0, t_buff, 0, recv);*

> *// Construct Packet*

> *MessagePacket packet = MessagePacket.BuildPacket(t_buff);*

> *// Send it via the Outbound Socket*

> *EnqueueOutboundMessage(packet);*

> }

>

> *Is there a way to avoid an ArrayCopy in this scenario? I have found that when receiving numerous small packets the lag time of the packets being proxied through the system is about 200ms, which is totally unacceptable for this server. Can anyone offer some insight on how I can improve its performance?*

>

> *~~K*

Re: Proxying bytes from one socket to another...