Design of high volume TCP Client

Asked 16 years, 2 months ago Modified 15 years, 11 months ago

Viewed 2k times

3

I have a .NET TCP Client that sends high volumes of messages to a (.NET async) TCP server.

I need to keep sending messages to the server but I run out of ports on the client due to TIME_WAIT.

How can a program continually and reliably send messages without using all of the available ports?

Is there a method to keep reusing the same socket. I have looked at Disconnect() and the REUSEADDRESS socket flag but cannot find any good examples of their use. In fact most sources say not to use Disconnect as it is for lower level use (i.e. it only recycles the socket handle).

I'm thinking that I need to switch to UDP or perhaps there is a method using C++ and IOCP?

tcp timeout client volume

Share

Improve this question

Follow

asked Oct 1, 2008 at 15:17

user21826

3,634 ● 5 ● 30 ● 28

6 Answers

Sorted by: Highest score (default)

5

You can keep the socket open if your server and client are aware of the format of the data. You're closing the socket so that the server can "see" that the client is "done".

If you have some protocol, then the server can "know" when it's finished receiving a block of data.

You can look for an End-of-message token of somekind, you can pass in the length of the message, and read the rest based on size, etc. Different ways of doing it.

But there's no reason to constantly open and close connections to the server -- that's what's killing you here.

Share Improve this answer Follow

answered Oct 1, 2008 at 18:18

Will Hartung

118k ● 20 ● 133 ● 207

1

Can your client just keep the same socket open and send messages in a loop?

```
open socket connection

while(running)
    send messages over socket

close socket connection
```

Share Improve this answer

Follow

answered Oct 1, 2008 at 15:24

Bill the Lizard

405k ● 211 ● 572 ● 889

1

TCP tries very hard to prevent congestion in the network. All new TCP connections begin in a "slow start" state, where they send only one packet and wait for an acknowledgement from the other end. If the ACK is received TCP will send two packets, then four, etc until it reaches its maximum window size.

If you are generating messages at high datarate, you really want to avoid opening and closing TCP connections. Every time you open a new connection you'll be back in slow start. If you can keep the socket open the TCP connection will get past the slow start state and be able to send data at a much higher rate.

To do this, you need to get the server to process more than one message on a connection (which means finding a way to delineate each message). If your server supports HTTP encoding of any sort this would work; make sure to examine any argument or configuration related to "persistent" connections or HTTP 1.1, because that is how HTTP sends multiple requests over a single TCP connection.

One option you mentioned is UDP. If you are generating messages at a reasonably high rate you're likely to lose some of them due to queues being full somewhere along the way. If the messages you are sending need to be reliable, UDP is probably not a good basis.

Share Improve this answer Follow

answered Oct 1, 2008 at 18:51

DGentry
16.3k ● 8 ● 53 ● 66

0

When coded like that it doesn't work.

The server only receives the first message.

When I open and close the socket then the server works but I run out of client ports.

I'm guessing it's the design of my server that is causing me to have to code the client like that.

So how does one code an Async server using .NET. I have followed the MSDN examples and numerous examples online.

Share Improve this answer Follow

answered Oct 1, 2008 at 15:49

user21826
3,634 ● 5 ● 30 ● 28

0

Would something along the lines of a message queuing service benefit your project? That way your client could pass as many messages to your server and your server could simply pull those messages from the queue when it can and as fast as your can and if you're client is sending more than it can handle, they'll simple enter the queue and wait to be processed.

Some quick Googling turned up this MSDN documentation on building a message queuing service with C#.

Share Improve this answer Follow

answered Oct 1, 2008 at 17:14

mwilliams
9,978 ● 13 ● 52 ● 71

0

The basic idea behind writing TCP servers (in any language) is to have one port open to listen for connections, then create new threads or processes to handle new connection requests.

```
open a server socket // this uses the port the
clients know about

while(running)
    client_socket = server_socket.listen
    fork(new handler_object(client_socket))
```

Here's a good example in C#.

Share Improve this answer Follow

answered Oct 1, 2008 at 18:13

Bill the Lizard
405k ● 211 ● 572 ● 889

Doesn't this scale very badly? Supposing you have 1000 connections. Are you suggesting that there should be 1000 threads? – Arafangion Oct 14, 2010 at 4:48

@Arafangion: If you expect that many concurrent connections then you can look into using thread pooling. That's used to set an upper limit on the number of threads that will be active at any time for your application, and helps manage their use. – Bill the Lizard Oct 14, 2010 at 11:16

Still seems high - an unnecessary overhead compared to select() or other polling. – Arafangion Oct 15, 2010 at 1:11