

Forum:[Hack Forums](#)

Thread: [eSock - Socket wrapper](#)

BahNahNah - May 22, 2015, 10:56 pm

Im still learning sockets so this is definitely not perfect.

Currently supports sending:

- uShort
- Integer
- String
- Byte array
- Boolean

Wrapper: [Pastebin](#)

Simple Usage

Server:

Code static void Main(string[] args)

```
{
    eSocket.Server _server = new eSocket.Server(1337);
    _server.DataRetrievedCallback += _server_DataRetrievedCallback;
    if (_server.Start())
        Console.WriteLine("Server started");
    else
        Console.WriteLine("Server failed to start");
    while (true)
        Console.ReadLine();
}
```

static void _server_DataRetrievedCallback(eSocket.Server.eSockClient c, eSocket.PacketStructure packet, eSocket.Server.PacketError er)

```
{
    Console.WriteLine("Message: {0}", new Message(packet.PacketBuffer).getMessage());
}
```

class Message : eSocket.PacketStructure

```
{
    public Message(string message)
        :base(message.Length, 0)
    {
        WriteString(4, message, Encoding.UTF8);
    }
    public Message(byte[] pack)
        :base(pack)
    {
    }
    public string getMessage()
    {
        return ReadString(4, Length - 4, Encoding.UTF8);
    }
}
```

```

    }
}
Client:
Code static void Main(string[] args)
{
    Console.WriteLine("Connecting...");
    eSocket.Client _client = new eSocket.Client();
    if (!_client.Connect("127.0.0.1", 1337))
    {
        Console.WriteLine("Failed to connect");
        Console.ReadKey();
        return;
    }
    Console.WriteLine("Connected!");
    while(true)
    {
        string inp = Console.ReadLine();
        _client.Send(new Message(inp));
    }
}

class Message : eSocket.PacketStructure
{
    public Message(string message)
        :base(message.Length, 0)
    {
        WriteString(4, message, Encoding.UTF8);
    }
    public Message(byte[] pack)
        :base(pack)
    { }
    public string getMessage()
    {
        return ReadString(4, Length - 4, Encoding.UTF8);
    }
}

```

ShittyAdvice - May 23, 2015, 7:55 am

I've always loved your work!

I just started so don't know too much :(, but noticed your ClientDisconnectCallback is null when catching (Socket > Server> RetrieveCallback)

This error is caused by disconnecting the client and having the server still open.

Adding a check fixes it, but might be something to look into

Code if(ClientDisconnectCallback != null)

ClientDisconnectCallback(new eSockClient(cSock), SE);

BahNahNah - May 23, 2015, 2:57 pm

(May 23, 2015 07:55 AM)Ras al Ghul Wrote: [CITE](#) I've always loved your work!

I just started so don't know too much :(, but noticed your ClientDisconnectCallback is null when catching (Socket > Server> RetrieveCallback)

This error is caused by disconnecting the client and having the server still open.

Adding a check fixes it, but might be something to look into

```
Code if(ClientDisconnectCallback != null)
```

```
ClientDisconnectCallback(new eSockClient(cSock), SE);
```

Thanks, i must have missed that one.

NETExplorer - May 23, 2015, 3:01 pm

(May 23, 2015 02:57 PM)BahNahNah Wrote: [\[CITE\]](#) Thanks, i must have missed that one.

You missed "that one"?

DEBUG MEMBERS DON'T MISS ANYTHING

I AM INPUTTING FOR YOUR IMMEDIATE REMOVAL

BahNahNah - May 23, 2015, 3:04 pm

(May 23, 2015 03:01 PM)PatPositron Wrote: [\[CITE\]](#) You missed "that one"?

DEBUG MEMBERS DON'T MISS ANYTHING

I AM INPUTTING FOR YOUR IMMEDIATE REMOVAL

Lucky i missed it before i was in debug xD

Nikey646 - May 23, 2015, 3:27 pm

You still have the problem related to the buffer being over-written if too many clients connect at once, as discussed [here](#)

An example implementation of a Socket Server which does not have this problem is shown in the [MSDN example](#) that shows how to use SocketAsyncEventArgs. (It uses the [BufferManager](#) class)

Serux - May 23, 2015, 3:49 pm

I refactored your code a bit and fixing some naming convention issues.

pastebin.com/VPi1RfJd

Although you are not doing anything with bufferSize in public Server(int port, int bufferSize = 1024, bool structured = true)

Did you just forget about it?

BahNahNah - May 23, 2015, 4:07 pm

(May 23, 2015 03:49 PM)Serux Wrote: [\[CITE\]](#) I refactored your code a bit and fixing some naming convention issues.

pastebin.com/VPi1RfJd

Although you are not doing anything with bufferSize in public Server(int port, int bufferSize = 1024, bool structured = true)

Did you just forget about it?

Yeah, they where initially split up into septate functions.

(May 23, 2015 03:27 PM)Nikey646 Wrote: [\[CITE\]](#) You still have the problem related to the buffer

being over-written if too many clients connect at once, as discussed [here](#)

-snip-

So for each new connection, the buffermanager needs to be recreated?

```
Code m_bufferManager = new BufferManager(receiveBufferSize * [b]numConnections[/b] * opsToPreAlloc, receiveBufferSize);
```

Is it a high possibility that the buffer will be overwritten?

possible solution?:

Make a new class that contains a socket and a buffer for each client.

Create a new instance of the class in the AcceptCallback for each new client that connects
pass this class through to the retrieveCallback, using a separate buffer for each client

Would this be a viable option?

Nikey646 - May 23, 2015, 5:20 pm

(May 23, 2015 04:07 PM)BahNahNah Wrote: [\[CITE\]](#) Yeah, they were initially split up into separate functions.

So for each new connection, the buffermanager needs to be recreated?

```
Code m_bufferManager = new BufferManager(receiveBufferSize * [b]numConnections[/b] * opsToPreAlloc, receiveBufferSize);
```

Is it a high possibility that the buffer will be overwritten?

possible solution?:

Make a new class that contains a socket and a buffer for each client.

Create a new instance of the class in the AcceptCallback for each new client that connects
pass this class through to the retrieveCallback, using a separate buffer for each client

Would this be a viable option?

I'm not sure how exactly the BufferManager works, but I do know when creating the BufferManager, you should do it once inside of the Server's Constructor, and the size of the BufferManager should be BufferSize * MaxConnections * Operations, with the last one being for Read/Write (so 2), and the second parameter will be how large the BufferSize should be. The downside of this approach is that a static number of clients can connect. So if the BufferManager only has room for 20 clients, then no more than 20 clients can be connected at once.

Personally, I think your suggested solution is the easier way to go, while still being maintainable, and in fact one of the other solutions MSDN has in an example is exactly that. As Shown [here](#).

BahNahNah - May 23, 2015, 8:47 pm

(May 23, 2015 05:20 PM)Nikey646 Wrote: [\[CITE\]](#) I'm not sure how exactly the BufferManager works, but I do know when creating the BufferManager, you should do it once inside of the Server's Constructor

-snip-

Thanks for the feedback.

I expanded the eSockClient class to have a buffer and an instance is now created when the server accepts a socket, so each socket has a separate buffer now.