

Forum:[Hack Forums](#)

Thread: [Differentiating Types Of Data \(Remote Desktop\)](#)

VelvetCoding - March 10, 2019, 4:35 pm

As I am nearing the development of my RAT application, I have stumbled into one final error. I have gotten my remote desktop system to work, however, the library i'm using for communication to the client only lets me send one byte array. This means that the server can only handle one type of data from the client at all times. In simple terms, when the remote desktop is active I can not call for other things from the client. Let me explain how this works:

1. The server issues a Start Remote Desktop command to the client that starts the remote desktop thread on the client and sets a bool on the server that tells the server all the of the next data it will receive will be an image.

2. The client enters the image streaming loop and sends images in byte form to the server. The server receives the bytes, but has no way of telling what the data is, because it does not have some sort of header that can be read in text form to tell it what it is. This is why I used a Boolean; So the server can tell it self that the data it is receiving is an image.

Main Problem To Solve: How can I feed the server information on what it is receiving in the data it self. (Which would allow me to send commands to the client, and get a response from the client all while sending the server images.)

Roadblock: The method that sends information to the server looks like this
(MainClient.Send(byte[]))

Help would be greatly appreciated. ?

gigajew - March 10, 2019, 4:39 pm

Use networkstream and write one byte header to tell whether it's an image or something else

VelvetCoding - March 10, 2019, 4:44 pm

(Mar 10, 2019 04:39 PM)gigajew Wrote: [\[CITE\]](#) Use networkstream and write one byte header to tell whether it's an image or something else

Are you able to provide an example?

gigajew - March 10, 2019, 5:06 pm

```
NetworkStream stream = new NetworkStream(Socket);  
stream.WriteByte(0x01);
```

This is also called the application layer in terms of networking, you need to figure out your own packet structures. You can use an existing serializer such as netserializer or manually parse your packets.

VelvetCoding - March 10, 2019, 10:48 pm

Problem Solved:

Client Side:

```
List<Byte> BytesToBeSent = new List<Byte>();
```

```
BytesToBeSent.Add(0); //Assuming 0 is the image type.
BytesToBeSent.AddRange(ImageByteArray);
MainClient.Send(BytesToBeSent.ToArray());
```

Server Side:

```
Byte[] ToProcess = ReceivedData.Skip(1).ToArray();
```

```
switch(ReceivedData[0])
{
    case 0: //Image type.
        //Do whatever you want with ToProcess .
        break;
    case 1: //Text type, for example, and so on.
        //...
        break;
    default:
        //...
        break;
}
```

gigajew - March 11, 2019, 4:07 am

That works but is probably very inefficient because of linq.

VelvetCoding - March 11, 2019, 11:48 am

(Mar 11, 2019 04:07 AM)gigajew Wrote: [\[CITE\]](#) That works but is probably very inefficient because of linq. It is inefficient, but I'm not doing anything super demanding on the server or client so its not too bad of a problem.

gigajew - March 11, 2019, 2:45 pm

Here's a more elaborated example. Not sure how to properly thread all the receive methods and such, but i think this is it. Hopefully this helps you some.

EDIT: Updated code

VelvetCoding - March 11, 2019, 3:58 pm

(Mar 11, 2019 02:45 PM)gigajew Wrote: [\[CITE\]](#) Here's a more elaborated example. Not sure how to properly thread all the receive methods and such, but i think this is it. Hopefully this helps you some. Thanks! This will help a lot. ?