

Forum:[Hack Forums](#)

Thread: [Capturing The Desktop Efficiently](#)

true - February 25, 2018, 9:10 pm

I've recently been working on a small RAT project, which will run on .NET 4.5. The goal of this project was to learn improve my knowledge of threading, TCP, and more. All has been going very well, but for the past couple of days I've been troubled by the stark contrast in remote desktop performance when compared with widely known RATs. Running it on my local network it goes at an acceptable speed(as expected, obviously), but when I use it on a computer off my local network, the speed is atrocious.

The way it works is the server will send the client a command to start sending the images. The client then begins doing so on a seperate thread, until it receives a stop command.

Below is my Task that sends the actual image, which is repeated in a loop. As you can see, I have the JPGCompression set almost as high as it can go, and I'm still experiencing a ton of lag. I'm not very knowledgeable when it comes to imaging, so this may be a silly issue. Hopefully someone can point me in the right direction, but I'm 99% sure this is somehow related to the encoding. I am using AForge for the Webcam, and I implemented it within a few minutes and it's incredibly fast, even on PC's on seperate networks. However, I would prefer not to use a 3rd party library for capturing the desktop.

abratar - February 25, 2018, 10:14 pm

I would check the current screen with the last one and see what has changed. If something changed then send that piece. If nothing has changed then you don't have to send anything because its the same. One example would be to slice your screen into 4/16/64 pieces so you don't have to send the whole screen but only the piece that has changed. whatever you prefer. But as i haven't worked with desktop capturing yet this might or might not be a good idea.

Ninja - February 26, 2018, 5:06 am

Wouldn't it be easier to copy the bitmap directly to the stream connecting the client and server instead of first into a memorystream and then to the server?

(Feb 25, 2018 11:14 PM)abratar Wrote: [CITE](#) I would check the current screen with the last one and see what has changed. If something changed then send that piece. If nothing has changed then you don't have to send anything because its the same. One example would be to slice your screen into 4/16/64 pieces so you don't have to send the whole screen but only the piece that has changed. whatever you prefer. But as i haven't worked with desktop capturing yet this might or might not be a good idea.

This. Or capture window events, like when a user moves a window, or a window changes, etc.

Ninja - February 26, 2018, 7:36 am

(Feb 26, 2018 07:45 AM)Pir?myd Wrote: [\[CITE\]](#) That definitely sounds like it would increase the speed a lot. How would I go about comparing the images? I recently found something called BitBlt which may be related to this, but I'm not sure how to use it

I don't know about the most efficient way, so don't take my word for it, but I am guessing either; comparing byte by byte and break when it differs, or getting the hash using a fast algorithm and comparing the hashes.

I just noticed the ProcessCommands function. Why not use an enum for that instead of a static Commands class?

Also, try to avoid public fields. Use properties instead.

chrizator - February 26, 2018, 8:43 am

(Feb 26, 2018 07:45 AM)Pir?myd Wrote: [\[CITE\]](#) How would I go about comparing the images? I once dealt with the same problem and I used unsafe code to compare the pixels of a bitmap (as integers) in order to work with pointers, since speed is crucial for those operations.

What I was doing is taking the previous screenshot and a new one and comparing the pixels for changes using simple XOR operations. This is actually a quite fast way of comparing 2 images.

```
Code public static void CompareImages(System.Drawing.Bitmap oldImage,
System.Drawing.Bitmap newImage)
{
    var lockRect = new Rectangle(Point.Empty, oldImage.Size);
    var oldData = oldImage.LockBits(lockRect, ImageLockMode.ReadWrite,
PixelFormat.Format32bppArgb);
    var newData = newImage.LockBits(lockRect, ImageLockMode.ReadOnly,
PixelFormat.Format32bppArgb);

    int pixLength = oldData.Height * Math.Abs(newData.Stride) / 4;

    unsafe
    {
        int* pOldPixels = (int*)oldData.Scan0;
        int* pNewPixels = (int*)newData.Scan0;

        for (int i = 0; i < pixLength; i++)
        {
            *pOldPixels++ ^= *pNewPixels++;
        }
    }

    oldImage.UnlockBits(oldData);
    newImage.UnlockBits(newData);
}
```

Don't be irritated by the names. The OLD image actually gets updated.

The bitmap, like arrays, is passed by reference. No need to return anything here.

If you receive the XORed image on the server side you can use the same method to compare the previously displayed image with the received one.

chrizator - February 26, 2018, 11:25 am

(Feb 26, 2018 11:34 AM)Pir?myd Wrote: [\[CITE\]](#) So if I'm understand correctly, the code you posted would be located on the client, and after the first screenshot, the client would then compare every

new screenshot to the previous one and then send the pixels that are different?

The code would be located on client AND server. The first image gets sent without any comparison and will be displayed on the server side (by a panel / picture box / whatever). The second screenshot will first of all be compared on the client-side to send the changed pixels only (less screenshot size -> less transfer time). As soon as the server received the changed pixels they will be compared with the previous image on the server-side. The same method can be used on both sides.

iNCEPTION_V - February 26, 2018, 1:49 pm

I would use ffmpeg or vlc, you can start the stream via cmd assuming the binary is there :)

Tsukiri - February 26, 2018, 1:52 pm

Interesting thread. I'd take a look at the source of QUASAR. It contains a pretty decent remote desktop and it is on github.

Also coded in the .NET framework.

Good luck!

chrizator - February 26, 2018, 2:18 pm

(Feb 26, 2018 02:52 PM)Tsukiri Wrote: [\[CITE\]](#) Interesting thread. I'd take a look at the source of QUASAR. It contains a pretty decent remote desktop and it is on github.

Also coded in the .NET framework.

Good luck!

Quasar is sending the whole image over the wire. Nothing sepcial there. But yes, it's interesting. ?

Tsukiri - February 26, 2018, 2:40 pm

(Feb 26, 2018 03:18 PM)chrizator Wrote: [\[CITE\]](#) Quasar is sending the whole image over the wire. Nothing sepcial there. But yes, it's interesting. ?

Putting all that time into making it efficient is probably not worth the time. In that case, I'd rather code a HVNC or HRDP module.

gigajew - February 26, 2018, 2:55 pm

I would personally define the Bitmap object and the Graphics object once, and re-use them.

Ninja - February 26, 2018, 3:40 pm

(Feb 26, 2018 03:40 PM)Tsukiri Wrote: [\[CITE\]](#) Putting all that time into making it efficient is probably not worth the time. In that case, I'd rather code a HVNC or HRDP module.

For a remote desktop it is well worth the time. The less bandwidth you use the better.

gigajew - February 26, 2018, 4:21 pm

Also, one of your culprits that I see right away are that you're calling `.ToArray()`;. I would probably stream images over a stand-alone socket and constantly write and update it.

cocaine_monkey - February 27, 2018, 2:25 am

How much base64 reduces file size?