# WCF replacement for cross process/machine communication

Asked 12 years, 7 months ago   Modified 7 years, 2 months ago

Viewed 3k times

9

I am working on a C# application that contains multiple windows services that will need to communicate with each other to pass data around. These services may be on the same machine but they could be remote. I looked into using WCF for this purpose but it seems like WCF is too heavy and has a lot of extra configuration that, to me, seems unnecessary (.NET 3.5 is a requirement here, I know that .NET 4 simplified this)

So my question is, what would be the best replacement to WCF, besides the deprecated .NET Remoting that provide this functionality?

c#   wcf

Share

Improve this question

Follow

edited Oct 8, 2017 at 14:13

Cœur

38.6k ● 26 ● 202 ● 276

asked May 15, 2012 at 20:24

Andrew Landsverk

673 ● 7 ● 17

Complaining about the configuration seems a tad petty. As for performance, NetTCP binding is available in 3.5, so not sure what your concern is there either. – Kirk Woll May 15, 2012 at 20:29

I was more curious about if there were alternatives and what they may be if any. Searching the Google always points me back to WCF. This is coming from someone who usually programs Java, so I'm used to RMI, EJB, etc. – Andrew Landsverk May 15, 2012 at 20:32

4 I reached more or less the same conclusion.I found that the configuration of WCF to be tricky and ill-documented, and under a thrashing,I also found it painfully slow and somewhat reluctant to release memory.Maybe I set it up wrong,but it was not as performant as I'd hoped, and the fix was not apparent in the myriad of configuration option.I ended up writing my own *lightweight* remoting framework which I hope one day to get approval to open-source.It was by no means trivial to implement. Unless you're butting into these kinds of problems,I'd stick with WCF until it *proves* to be problematic. – spender May 15, 2012 at 20:37

## 3 Answers

Sorted by: Highest score (default)

16

I have been using PInvoke to access the Windows RPC runtime for nearly 8 years. It's wicked fast and very reliable as far as a transport goes. When combined with a fast serializer like protobuf-csharp-port the resulting communications are rock solid and very fast.

So to build this from the ground-up this requires three parts:

1. Google's Protocol Buffers (protobuf-csharp-port) for serialization.
2. My own CSharpTest.Net.RpcLibrary for the transport.
3. A bit of glue code to put them together from protobuf-csharp-rpc.

These are all available on NuGet in the following packages: Google.ProtocolBuffers, CSharpTest.Net.RpcLibrary, and Google.ProtocolBuffers.Rpc.

The following is a quick run-down on getting started:

1. define a set of messages and a service using the Google Protocol Buffer Language.
2. Once you have that defined you will run ProtoGen.exe to generate the service stubs and messages in C#. Be sure to add the "-service_generator_type=IRPCDISPATCH" to generate the correct service code.
3. Now that you have the generated source files add them to a project and reference the three assemblies from the packages listed above.
4. Lastly take a look at the sample client/server code on the protobuf-csharp-rpc project page. Replace the "SearchService" with your service name, and you should be ready to run.
5. Optionally change the configuration of the RPC client/server. The example shows the use of LRPC

which is local-host only; however the DemoRpcLibrary.cs source file show TCP/IP and Named Pipes as well.

You can always email me (roger @ my user name) for any further information or examples.

**Update**

I wrote a quick startup guide: WCF replacement for cross process/machine communication.

Share Improve this answer Follow

edited May 16, 2012 at 1:07

answered May 15, 2012 at 21:02

csharptest.net
**64.1k** ● 11 ● 76 ● 91

This looks like the alternative I was looking for, Thanks for the info! – Andrew Landsverk May 15, 2012 at 21:33

1 @Andrew Landsverk, see also: csharptest.net/1177/… – csharptest.net May 16, 2012 at 1:08

1 Hah! Your post confirms everything I suspected about WCF but never got round to measuring. Your solution is very interesting... Definitely an answer worthy of more than +1. – spender May 16, 2012 at 10:04

Thank you very much. Very helpful. Your log4net XSD is also very helpful. So thanks for that too. – One-One May 28, 2012 at 12:16

@csharptest.net Thank you for sharing the libray, great work. I am trying to figure out how to deal with long-running operations. The Execute() call is synchronous and theoretically I could refactor the library to split the call/return sequence. Is there a built-in method of "eventing" or server -> client calls? I looked through "Microsoft RPC Programming Guide" by Shirley and Rosenberry, couldn't find
– Yuriy Gettya Aug 11, 2015 at 11:50

2

You may want to look into ZeroMQ, it's very lightweight and effective and comes with good C# bindings. (Typing this on my mobile so you'll have to google for it yourself for now, sorry).

Share Improve this answer Follow

edited May 14, 2014 at 14:34

chollida
7,894 ● 12 ● 57 ● 86

answered May 15, 2012 at 20:53

gjvdkamp
10.5k ● 4 ● 39 ● 53

I wouldn't exactly call ZeroMQ lightweight. – Erik Forbes May 14, 2013 at 20:34

ZeroMQ is very good provided you don't care for a response and aren't worried about messages getting dropped on the floor. Like all queues, only so much backlog can be stored. ZeroMQ's performance in guaranteed delivery and request/response configurations is particularly bad.
– csharptest.net Nov 5, 2015 at 22:13

0

Look at NFX Glue.

It is way faster than WCF for coupled systems.

Interprocess communication with Glue Blog

Benchmark

Code: https://github.com/aumcode/nfx

Share Improve this answer Follow

answered May 5, 2015 at 17:16

itadapter DKh
590 ● 3 ● 7

where is documentations? – Ersin Tarhan Mar 20, 2016 at 17:48