

Re: client –server interaction over XML supporting multiple protocols

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2005-03/0240.html

From: David (FlyLikeAnEagle_at_United.Com)

Date: 03/02/05

Date: Wed, 02 Mar 2005 04:52:23 GMT

On Tue, 1 Mar 2005 05:54:12 UTC, "rdh" <rohit.dhamija@gmail.com> wrote:

> *Hi all,*
>
> *I am in process of developing a Server in C++ supporting multiple*
> *protocols. The server will be exposing various functionalities, and the*
> *clients can communicate over any of the protocols may be TCP, IPX, SAP,*
> *NETBEUI to access the server to access the functionalities exposed. The*
> *server doesnot know in advance which client is using what protocol.*
> *ALSO, ALL THE INTERACTION WILL BE MADE OVER XML.*
>
> *example my server has functionality X()*
>
> *and i have n number of client.*
> *client1 is communicating over protocol TCP to access X()*
> *client2 is communicating over protocol IPX to access X()*
>
> *..*
> *and so on.*
>
>
> *We had already developed (prepared a rough code sketch), a server that*
> *is able to handle multiple clients at the same time over TCP/IP. But*
> *now, we need to enhance the same so that it can intearct with clients*
> *irrespective of the protocol being used.*
>
> *I am struck with following issue:*
>
> *1) How to make the client–server flow so that the server knows from*
> *which protocol the client has communicated so that the server can send*
> *the reply over the same protocol and this whole process is*
> *multithreaded.*
>
> *What all steps are needed to take care of while building the system*
> *most efficient.*
>

- > *In summary*
- > *need to make server in such a way that it may accept a request from*
- > *connection oriented protocol as well as connection-less protocol at the*
- > *same time. so need to develop a mechanism for the same the work for all*
- > *conditions.*
- >
- > *Please suggest. Also please feel free to send your comments/suggestions*
- > *to make this system more efficient.*
- >
- > *All suggestions are welcome.*
- >
- > *Thanks,*
- > *rdh*

Hello rdh,

Look up some of the ancient examples of server code under Unix/linux for TCP/IP. The typically most complicated example of how to write the server side covers multi-service, multi-protocol servers.

You simply need to associate the proper protocol with each of your connections. This may already be part of the connection information if all of your protocols come in on a standard interface. If you use multiple threads or interfaces to handle all of the protocols you just need to provide a common interface that can be used by the rest of your request-response engine.

You will need to decide if your target operating system, server model, and request-response engine is best implemented as a single process or multiple processes. 'process' here could mean just about anything based on the various operating system models that are available. For instance, the overhead of your XML decoder-encoder or request-response engine for each conversation may overload some operating system models. At times it is better to keep some services separate. This all depends on the complexity of your function X().

It is likely that the stream oriented (TCP) and packet oriented (UDP) protocols will provide a unified view to another layer of your server that actually performs the XML parse or encode. The complexity of putting this all together also largely depends on your experience and the tools that you choose for the project. If you doubt your ability to put it all together the first time, implement the final optimized strategy in steps. That also allows you to change your design if problems are found in the design.

You may be able to find much of what you need already available in other packages or combinations of packages. I have my own client-server models that I can simply drop just the conversation engine (request-response engine above) in and select what kind of server or client model I want to implement. I'd offer them but

they are not likely to meet your needs as they target a certain class of linux and other embedded systems.

In review, for (1) if your communication model is Unix/linux based or Microsoft based, check out the definition of communications sockets. You are likely to find that the definition of a socket can be either UDP, TCP, IPX, and so on. If I recall correctly these are called the Interface Families. You just need a plug-in or support for all of your desired protocols in your operating system and the socket communications layer will hide most of the details for you. You will still need to address the stream vs. packet mode yourself.

The optimized solution (2) for the entire multi-protocol server, XML, and function X() are best derived based on experience. The size of the XML and X() functionality will determine the demands on your operating system. C++ can easily handle multi-protocol servers and the other components you are likely to need. However, some of the underlying complexity of the XML engine or X() functionality may degrade performance as the number of conversations rise. This generally isn't a problem for simple communications and X() functionality. If X() was a time consuming or large overhead task, your performance may be better with a slightly different model. You will need to play around with the design if you cannot meet your target goals. You likely have some idea of the complexity of the XML and X() functionality already. Also consider the number of simultaneous clients served, client-to-client interactions if needed, and the desired response time.

David