- Living in São Paulo, Brazil

- Interesting in
* NFL, NBA
* Lockon, Black Shark
* Dream Theater, drumming

- Developing since 1994 with
* Clipper (Summer '87 and 5.02)
* FoxPro (DOS, 2.6), Visual Foxpro (6, 7, 8, 9)
* Delphi (1, 2, 5, 7)
* C# (1.1, 2.0)

Article Top

Comments and Discussions

Hint: For improved responsiveness ensure Javascript is enabled and choose 'Normal' from the Layout dropdown and hit 'Update'.
You must Sign In to use this message board.

Search this forum Go

Profile popups Spacing Relaxed Noise Very High Layout Normal Per page 10 Update

View All Threads First Prev Next

Any upgrade of ALAZ planned? Or even available soon? mklaey 7:21 31 May '12

Hi Andre Luis,

I am using your excellent socket extension in another open source project called "YAT - Yet Another Terminal" (https://sourceforge.net/projects/y-a-terminal/). YAT features an RS-232/422/423/485 terminal optimized for communication with embedded systems. In addition it also supports TCP client/server/AutoSocket and UDP connections for PC based embedded systems simulations, AutoSocket being an automatic client/server detection mechanism.

At first I was using ALAZ 1.4/1.5 on .NET 2.0, then updated to ALAZ 2.0 and .NET 3.5. After the update it didn't build right from start. But then I realized that back in 2007 I made significant modifications to ALAZ:
- Adding support for UDP (HostType.htUdp)
- Improving stability by adding exception handling at several places
- Changing SocketsEx\BaseSocketConnectionHost.StopConnections() from blocking to non-blocking
- Some other modification which in retrospect where rather silly

I was glad to see that you have improved the exception handling in ALAZ 2.0. So I didn't need to redo these modifications. But I had to redo the support for UDP. Isn't a big thing, still, I am wondering whether you could consider to add UDP support to the original ALAZ library for future versions. In retrospect, I should have given you this input already in 2007, my fault. Better later than never

UDP
In order to support UDP, I made modifications at the following locations:
- \SocketsEx\SocketConnector
> private ProtocolType FProtocolType;
> Constructor taking the protocol type as additional argument
> BeginConnect() taking the protocol type into account
- \SocketsEx\SocketClient
> using System.Net.Sockets;
> Another constructor taking the protocol type as additional argument
- \SocketsEx\SocketServer
> using System.Net.Sockets;
> Another constructor taking the protocol type as additional argument
- \SocketsEx\SocketClientSync
> private ProtocolType FProtocolType;
> Constructor taking the protocol type as additional argument
> Connect() taking the protocol type into account
- \SocketsEx\BaseSocketConnectionHost
> private ProtocolType FProtocolType;
> public ProtocolType ProtocolType { get; }
> Constructor: Protocol type as additional argument

SocketsEx\BaseSocketConnectionHost.StopConnections()
I am not exactly getting why this method is blocking and therefore makes Stop() blocking. The Start() method isn't blocking.
If Stop() is called from a GUI thread and the GUI is attached to the Disconnected event, a dead-lock happens:
- The GUI thread is blocked here
- FireOnDisconnected is blocked when trying to synchronize Invoke() onto the GUI thread

However, I am then getting an ObjectDisposedException in CloseConnection() on connection.Socket.Shutdown(SocketShutdown.Send) when stopping:
System.ObjectDisposedException was unhandled
Message="Cannot access disposed object \"System.Net.Sockets.Socket\"."
Source="System"
ObjectName="System.Net.Sockets.Socket"
StackTrace:
at System.Net.Sockets.Socket.Shutdown(SocketShutdown how)

at ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.CloseConnection(BaseSocketConnection connection) in \SocketsEx
\BaseSocketConnectionHost.cs:Line 1805.
at ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginDisconnectCallbackAsync(Object sender, SocketAsyncEventArgs e) in \SocketsEx
\BaseSocketConnectionHost.cs:Line1489.
at System.Net.Sockets.SocketAsyncEventArgs.OnCompleted(SocketAsyncEventArgs e)
at System.Net.Sockets.SocketAsyncEventArgs.ExecutionCallback(Object ignored)
at System.Threading.ExecutionContext.runTryCode(Object userData)
at System.Runtime.CompilerServices.RuntimeHelpers.ExecuteCodeWithGuaranteedCleanup(TryCode code, CleanupCode backoutCode, Object userData)
at System.Threading.ExecutionContext.RunInternal(ExecutionContext executionContext, ContextCallback callback, Object state)
at System.Threading.ExecutionContext.Run(ExecutionContext executionContext, ContextCallback callback, Object state)
at System.Net.Sockets.SocketAsyncEventArgs.FinishOperationSuccess(SocketError socketError, Int32 bytesTransferred, SocketFlags flags)
at System.Net.Sockets.SocketAsyncEventArgs.CompletionPortCallback(UInt32 errorCode, UInt32 numBytes, NativeOverlapped* nativeOverlapped)
at System.Threading._IOCompletionCallback.PerformIOCompletionCallback(UInt32 errorCode, UInt32 numBytes, NativeOverlapped* pOVERLAP)
I simply commented-out the connection.Socket.Shutdown(SocketShutdown.Send) call and it works in my case (but probably not in other cases, also see below).

Disposal
Probably due to the modifications mentioned above, tt seems that TCP server sockets don't properly shut down when Dispose() is called. As I understand the purpose of Dispose(), a call to this method must always be possible and it must immediately release all resources the object holds. In case of TCP server this seems not the case.

Distribution
There are a few files in your distribution which I think shouldn't be in there. I had to remove/untick them before checking them into my SVN.
- *.user
- \Code\Demos\Echo\EchoConsoleServer\Service References
- \Code\Demos\Echo\EchoFormServer*.tmp
- \Code\Source\ALAZ.SystemEx\bin\
- \Code\Source\ALAZ.SystemEx.NetEx\bin\

Some other inputs
- ALAZ AssemblyInfo.cs should have [assembly: CLSCompliant(true)] enabled
> Allows libraries/programs to be CLS compliant
- The "ALAZLibSN.snk" should be within "\Properties"
- The "ALAZLibSN.snk" should be available in both projects
- The "ALAZLibSN.snk" should be referenced/enabled in both projects
- How about adding a static class diagram to the Visual Studio project?
- How about providing BaseSocketConnectionHost/SocketClient/SocketServer constructors that use ctWorkerThread by default?
- How about using System.Threading.Timeout.Infinite to emphasize infinite timeout values?
- How about providing default constants for 1024 * 2 and 1024 * 16?

I came across another issue when shutting down a pair of an ALAZ client connected to an ALAZ server within the same application. I ran into a deadlock while shutting down the open connection. It happens in ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.Active.get(). Is there a need to lock the field when reading its status? The situation is as follows:
Thread A) at ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginReadCallbackAsyncP trying to call connection.BeginDisconnect()
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.Active.get() Line 286
>
ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.FireOnException(ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection
connection =
{ALAZ.SystemEx.NetEx.SocketsEx.ClientSocketConnection}, System.Exception ex = {Cannot evaluate expression because the current thread is in a sleep, wait, or
join}) Line 551
>
ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginDisconnect(ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection
connection =
{ALAZ.SystemEx.NetEx.SocketsEx.ClientSocketConnection}) Line 1453
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.BeginDisconnect() Line 558
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginReadCallbackAsyncP(object state =
{System.Net.Sockets.SocketAsyncEventArgs}) Line 1161
Thread B) at ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginReadCallbackAsyncP trying to get the connection state
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.Active.get() Line 286
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginReadCallbackAsyncP(object state =
{System.Net.Sockets.SocketAsyncEventArgs}) Line 1146
The Main Thread at ALAZ.SystemEx.NetEx.SocketsEx.SocketServer.Stop()
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.Active.get() Line 286
>
ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.BeginDisconnect(ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection
connection =
{ALAZ.SystemEx.NetEx.SocketsEx.ServerSocketConnection}) Line 1435
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnection.BeginDisconnect() Line 558
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.StopConnections() Line 340
> ALAZ.SystemEx.NetEx.dll!ALAZ.SystemEx.NetEx.SocketsEx.SocketServer.Stop() Line 207

Also, there seems to be a null reference issue in ALAZ.SystemEx.NetEx.SocketsEx.BaseSocketConnectionHost.StopCreators() on line 311 (FWaitCreatorsDisposing.WaitOne). I added a try-catch around the statement.

Another issue I found in SocketListener.BeginAcceptCallback on line #229 AcceptAsync(e2):

An unhandled System.ObjectDisposedException was thrown while executing this test : The SafeHandle has been closed.

at System.Net.Sockets.Socket.AcceptAsync(SocketAsyncEventArgs e)
at ALAZ.SystemEx.NetEx.SocketsEx.SocketListener.BeginAcceptCallback(Object state) in D:\Sandboxes\YAT_Trunk\ALAZ\Source\ALAZ.SystemEx.NetEx
\SocketsEx\SocketListener.cs:Zeile 229.
at System.Threading._ThreadPoolWaitCallback.WaitCallback_Context(Object state)
at System.Threading.ExecutionContext.runTryCode(Object userData)
at System.Runtime.CompilerServices.RuntimeHelpers.ExecuteCodeWithGuaranteedCleanup(TryCode code, CleanupCode backoutCode, Object userData)
at System.Threading.ExecutionContext.RunInternal(ExecutionContext executionContext, ContextCallback callback, Object state)
at System.Threading.ExecutionContext.Run(ExecutionContext executionContext, ContextCallback callback, Object state)
at System.Threading._ThreadPoolWaitCallback.PerformWaitCallbackInternal(_ThreadPoolWaitCallback tpWaitCallBack)
at System.Threading._ThreadPoolWaitCallback.PerformWaitCallback(Object state)

This issue occured while performing endurance tests using YAT AutoSockets.

And the last issue for today, found in BaseSocketConnection.Active:

I get another deadlock because I have to synchronize events onto my main/GUI thread. In case of having an application with two sockets connected to each other I get a deadlock upon shut down. Thus I have removed the lock in the get{} property of BaseSocketConnection.Active.

UDP
I have removed all UDP stuff and use System.Net.Sockets.UdpClient directly

Now, are you still developing ALAZ? Or has this project, as so many others, come to a dead-end?

Best regards,
Matthias

Sign In·Permalink

Re: Any upgrade of ALAZ planned? Or even available soon? **Andre Azevedo** 8:14 31 May '12

Last Visit: 18:00 31 Dec '99 Last Update: 7:59 11 Oct '12 Refresh 1

General News Suggestion Question Bug Answer Joke Rant Admin