VB and Winsock, the Basics

This introductory text assumes you have a good knowledge of Visual Basic and the Basic language. This is also a supplement to the ENGR 3553 Networking Laboratory #3. So, fire up VB and let’s get started.

First we have to load the Winsock component on your Component Toolbar. Simply place the mouse cursor on the Component Toolbar and press the right mouse button. Select Add Component. Slide down the list until you reach Microsoft Winsock and check the box. Once you click OK or Apply the Winsock control appears on the Toolbar. Now we can highlight the control on the toolbar and draw the component on the Form. We should now rename the control and I usually provide a LocalPort property so when the Form initially loads we can simply put the Winsock control into Listen mode if a Server is required. For Client application you may want to specify the port you wish to use along with the IP or Hostname of the Server you wish to connect to.

Private Form_Load()

    sckServer.Listen

End Sub

This is all you need to start your Winsock control, that we have named sckServer, in server mode. Again, We must have specified a LocalPort property at design time else we would have to add the line

    sckServer.LocalPort = 567

Or whatever port number we wish to listen to. This gives us

Private Form_Load()

    sckServer.LocalPort = 567
    sckServer.Listen

End Sub

For a client application we may have two Editboxes on our Form, one for the ip or hostname and the other for port value. For this example the editboxes will be called ipedit and portedit. We will also have a command button named connectbutton. When a click on the connectbutton occurs and we want to connect to our server the following code should do the trick. Our Winsock control should be named sckClient. You should not specify a value for LocalPort for a client application even though it may seem to work fine I will show later that this is a no-no.

Private connectbutton_Click()

    sckClient.RemoteHost = ipedit.text
    sckClient.RemotePort = Val(portedit)
    sckClient.Connect

End Sub
What I like to do is have dual function on the connectbutton, meaning if sckClient is connected then it will disconnect and if sckClient is disconnected then connect. Like this

Private connectbutton_Click()

    If sckClient.State <> sckConnected Then
        sckClient.RemoteHost = ipedit.text
        sckClient.RemotePort = Val(portedit)
        sckClient.Connect
        connectbutton.Caption = “Disconnect”
    Else
        sckClient.Close
        connectbutton.Caption = “Connect”
    End If

End Sub

The Client example is what will work for you all the time. The Server may require multiple connections to communicate to more than one client at a time. For this case see “Multiple Connections to a Server Socket using VB and Winsock”.
**VB and Winsock, Sending and Receiving Data**

To send and receive data using Winsock you would use the SendData and GetData method of Winsock to provide these functions.

The SendData method is an indiscriminate procedure that sends the data across the network without caring what data type it is, with one exception Variant data. Simply once connected we would say

```
sckClient.SendData varname
```

where varname is the variable in which the data to be sent is contained. To monitor the progress of the transmission there are two ways to see how things are going or what is required to do next. The SendProgress event occurs while you are sending data. There are two two variables bytesSent and bytesRemaining. These variables tell how much data has been sent and how many bytes left in the send buffer. The SendComplete event tell us when successful transmission of the data occurs.

To illustrate these events our Form will have a StatusBar called status, a Command button called sendbutton, a TextBox called sendtext, and our Winsock control sckClient.

```vbnet
Private Sub sendbutton_Click()
    sckClient.SendData sendtext.text
End Sub

Private Sub sckClient_SendProgress(bytesSent as Long, bytesRemaining as Long)
    Status.SimpleText = Str$(bytesSent) & " bytes sent and " & Str$(bytesRemaining) & " to go"
End Sub

Private Sub sckClient_SendComplete()
    Status.SimpleText = "Send Complete"
End Sub
```

If your text in the sendtext textbox is not very long you may not see the SendProgress event updating the statusbar but it is working.

To receive data you must specify the type of data to be received. By default string data is to be received, however, you can tell what type of data is to be received. Also, one thing you must be aware of is the size of your receive buffer. What happens if you are receiving a large block of data the DataArrival Event provides you a variable bytesTotal which tells how much data is being received by this event. bytesTotal is misleading in that it may not mean the total number of bytes being transmitted. If bytesTotal equals your receive buffer size it is more than likely that there is still more of the same transmission still to come. This will raise another DataArrival event to continue the transmission.
The first example shows reception of different data types. The one data type not able to transferred is the Variant data type which can be handled by either converting to a double real number or thrown into a TextBox and transferred as those types.

Private Sub sckClient_DataArrival(bytesTotal as Long)

    Dim sdata as string
    Dim idata as Integer
    Dim ddata as Double

    sckClient.GetData sdata
    sckClient.GetData idata, vbInteger
    sckClient.GetData ddata, vbDouble

End Sub

In the GetData sdata you could put vbString at the end without effect. Also this example just shows how you would enter the GetData commands this would not work if you were to use this code specifically.

In order to handle multiple buffers of received data initialize a counter on Form_Load or on a connection and increment the counter on every visit to DataArrival then the actual total number of bytes received from the complete transmission would be

    completebytes = completebytes + bytesTotal

Then once the SendComplete event occurs do what ever you want with the variable completebytes then reinitialize it to 0 so it can be reused the next time data is received. There are other ways to deal with multiple calls to DataArrival from one transmission this is just one possibility.
Multiple Connections to a Server Socket Using VB and Winsock

First we must learn the rules about making a Control Array. The Winsock usercontrol, as with other usercontrols, can be set to have multiple instances that are arranged in a Control Array. At Design Time, the Index property of the control must be set to zero (0). If the control Index property is left as NULL, the default value, multiple instances of the control are not allowed. Now, when we require another instance of the control we can simply load a new instance and show if required.

Once we have our winsock control (sckServer) placed on our form and set the Index property to zero, when the form is loaded we can set sckServer(0) into Listen, or if you wish server mode. Be sure that the port number you wish to use is specified in the LocalPort property at some point.

```vba
Private Sub Form_Load()
    sckServer(0).LocalPort = 567
    If sckServer(0).State <> sckClosed Then sckServer(0).Close
    sckServer(0).Listen
End Sub
```

Then once a ConnectionRequest Event occurs, when a client is trying to connect to our server application, we simple create a new instance of the Winsock Control and connect to the client using it. The following code shows the creation of sckServer(1). In most applications a variable would be used to set the Index value of the control instance.

```vba
Private Sub sckServer_ConnectionRequest(Index As Integer, ByVal requestID As Long)
    If Index = 0 Then
        Load sckServer(1)
        sckServer(1).Accept requestID
    End If
End Sub
```

We see that the request for connection to the server is made to sckServer(0), hence we check and make sure the client is trying to connect on the 0 instance. If so we then load a new instance of sckServer and accept the connection request on the new instance.

Consequently, when the client wants to disconnect you should close the instance of Winsock and unload the instance so it may be used again.

```vba
Private Sub sckServer_Close(Index As Integer)
    sckServer(1).Close
    Unload sckServer(1)
End Sub
```
The Flexibility of Winsock

I am going to show how we can use a single Winsock component to provide both server and client functionality within the same application.

We first start with our Winsock component sckdual. We then, as before set the socket in Listen mode on Form load. We will need multiple instances so the Index property should be set to 0 and a LocalPort should be assigned. Index must be changed at design time in the Property Window.

Private Sub Form_Load()
    sckdual.LocalPort = 567
    sckdual.Listen
End Sub

For each new instance created to serve a client connection a LocalPort should be specified. To use sckdual as a client to connect you must specify a RemotePort and a RemoteHost in order to complete a client connection. For a client usage LocalPort should be 0 or else an address in use error will occur. Here is a simplified code for a connection request from a client so we want to provide the server functionality.

Private Sub sckdual_ConnectionRequest(Index as Integer, ByVal requestID as Long)
    If Index = 0 then
        Load sckdual(1)
        sckdual(1).LocalPort = 567
        sckdual(1).Accept requestID
    End If
End Sub

If we want to connect to another server we must give client functionality to sckdual. We will have a command button called connectbutton and we will hard code our remote info.

Private Sub connectbutton_Click()
    Load sckdual(2)
    sckdual(2).LocalPort = 0
    sckdual(2).RemotePort = 567
    sckdual(2).RemoteHost = "216.211.76.199"
    sckdual(2).Connect
End Sub
This program is made to show how Winsock is used with VB

The program allows 5 connections to the server and has
client function to connect to other chatters.

Written by Bruce Misner 2001

carray tells if there is a connection on that server
instance

Dim carray(5) As Boolean

Connect button click event handler

Private Sub connectbutton_Click()

' Test to see if you want to connect or disconnect from a server

If connectbutton.Caption = "Connect" Then

' load new server instance of sckServer
'sckServer(6) is for client use

Load sckServer(6)
sckServer(6).LocalPort = 0
sckServer(6).RemoteHost = rhosttext.Text
sckServer(6).RemotePort = Val(rporttext.Text)
sckServer(6).Connect
connectbutton.Caption = "Disconnect"
Else

' to disconnect unload the sckServer instance

sckServer(6).Close
connectbutton.Caption = "Connect"
Unload sckServer(6)
End If

End Sub

This allows the user to select a connected client
to send messages to

Private Sub connects_Click()

' takes select ip from combobox and puts it in
'the send to textbox

If sendto.Text <> connects.Text Then sendto.Text = connects.Text

End Sub
Private Sub exitbutton_Click()
    If sckServer(0).State <> sckClosed Then sckServer(0).Close
    Unload Me
End Sub

Private Sub Form_Load()
    If sckServer(0).State <> sckClosed Then sckServer(0).Close
    sckServer(0).LocalPort = 567
    sckServer(0).Listen
    localname.Caption = sckServer(0).LocalHostName
    localprt.Text = Str$(sckServer(0).LocalPort)
    For i = 1 To 5
        carray(i) = False
        connects.AddItem (Str$(i) & "-"
    Next i
End Sub

Private Sub sckServer_Close(Index As Integer)
    carray(Index) = False
    If connects.List(Index - 1) = sendto.Text Then sendto.Text = ""
    connects.List(Index - 1) = Str$(Index) & ":"
    sckServer(Index).Close
    While sckServer(Index).State <> sckClosed
        Wend
    'MsgBox "ready to unload " & Str$(Index)
    Unload sckServer(Index)
End Sub
When a client tries to connect to our server, load a new instance of `sckServer` and update connection info.

```vbscript
Private Sub sckServer_ConnectionRequest(Index As Integer, ByVal requestID As Long)
    If Index = 0 Then
        For i = 1 To 5
            If Not (carray(i)) Then
                Load sckServer(i)
                sckServer(i).Accept requestID
                carray(i) = True
            connects.List(i - 1) = Str$(i) & "-" & (sckServer(i).RemoteHostIP)
        GoTo foundindex
        End If
    Next i
    End If
foundindex:
End Sub
```

When data arrives from other chat applications put message in textbox and display who sent the message.

```vbscript
Private Sub sckServer_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    Dim rdata As String

    sckServer(Index).GetData rdata
    receivetext.Text = rdata
    fiplabel.Caption = sckServer(Index).RemoteHostIP

End Sub
```

If a socket error occurs close the socket.

```vbscript
Private Sub sckServer_Error(Index As Integer, ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
    If sckServer(Index).State <> sckClosed Then sckServer(Index).Close
    Unload sckServer(Index)
    MsgBox "socket error"
End Sub
```
' Sends message to other chatter that you have selected
' from the connection list.
'
Private Sub sendbutton_Click()

    sckServer(Val(Mid$(sendto.Text, InStr(sendto.Text, "-")) - 1, 1)).SendData sendtext.Text

    sendtext.Text = ""

End Sub