

Forum:[Hack Forums](#)

Thread: [how to send file with progressbar via TcpClient ?](#)

elmagek - October 6, 2021, 4:48 am

Code Dim ns As NetworkStream

ns = cli.GetStream

Dim bytes() As Byte = New Byte(cli.ReceiveBufferSize - 1) {}

Dim i As Integer = 0

Do While cli.GetStream.DataAvailable = True

bytes(i) = cli.GetStream.ReadByte()

i += 1

'msg = System.Text.Encoding.ASCII.GetString(bytes, 0, i)

'Form3.ProgressBar1.Value = cli.Available / i * 100

Loop

i will send by client to server

server and client work on my pc for test

so i want send file by client

and received it by server and progressbar will be in server form

Donkey Death Jam - October 6, 2021, 6:54 am

(Oct 6, 2021 04:48 AM)elmagek Wrote: [\[CITE\]](#) Code Dim ns As NetworkStream

ns = cli.GetStream

Dim bytes() As Byte = New Byte(cli.ReceiveBufferSize - 1) {}

Dim i As Integer = 0

Do While cli.GetStream.DataAvailable = True

bytes(i) = cli.GetStream.ReadByte()

i += 1

'msg = System.Text.Encoding.ASCII.GetString(bytes, 0, i)

'Form3.ProgressBar1.Value = cli.Available / i * 100

Loop

i will send by client to server

server and client work on my pc for test

so i want send file by client

and received it by server and progressbar will be in server form

The way you're setting the progress bar is mathematically flawed: typically, to get the progress, the formula is: $\text{currentProgress} / \text{maximumProgress} * 100$. In your case, you are constantly changing the maximumProgress (integer i). 'cli.Available' is only going to return what you have currently received on your network card - it doesn't account for things such as network propagation. TCP is sent in data packets, the packets are sent as a stream across the network which are ultimately received as a stream on your end. Because of this, the length of the message is not known unless you either specify it in your message initially, or you finish the communication - at which point there would be no point of a progress bar.

You want to establish a handshake which will exchange the length of the file in bytes, then you can adjust your code to read the file size at the start of RX and use this value for the maximumProgress variable. You could always make the first four bytes receive the length of the file (assuming you don't want to transact any file size past the range of unsigned int) - ideally send this in network byte order (big-endian).

Here's a StackOverflow post which might help you out:

<https://stackoverflow.com/questions/1339270/how-do-i-prefix-the-length-of-message-in-tcp-ip>

elimagek - October 6, 2021, 8:54 am

can you help with sample source ?

Donkey Death Jam - October 6, 2021, 1:05 pm

Here's a very crummy example I've whipped up:

Code Imports System

Imports System.Net

Imports System.Net.Sockets

Imports System.Security.Cryptography

Imports System.Threading

Module Program

''' <summary>

''' Boolean representing whether the server is running/stopped

''' </summary>

Dim serviceRunning As Boolean = False

''' <summary>

''' Abstract representation of our client

''' </summary>

Private Structure Client

''' <summary>

''' Socket instance belonging to the client

''' </summary>

Public socket As Socket

''' <summary>

''' Holds any received data from a client socket

''' </summary>

Public serviceBuffer() As Byte

''' <summary>

''' Used to ensure correct content is being processed as a single message

''' </summary>

Public isProcessing As Boolean

''' <summary>

''' Length of the message we are processing

''' </summary>

Public messageLength As UInteger

''' <summary>

''' Bytes we've currently read

''' </summary>

Public bytesRead As UInteger

''' <summary>

```

''' Initialise the structure properties
''' </summary>
''' <param name="bufferSize">size of receive buffer </param>
Public Sub New(sock As Socket, bufferSize As Integer)
    socket = sock
    ReDim serviceBuffer(bufferSize)
    isProcessing = False
End Sub
End Structure

''' <summary>
''' Initial entry point
''' </summary>
''' <param name="args">Launch arguments</param>
Sub Main(args As String())

    Console.WriteLine("[SERVER] Initiating server...")

    ' Spin a server on an isolated thread
    Dim serviceWorker As New Thread(AddressOf Service)
    serviceWorker.IsBackground = True
    serviceWorker.Start()

    ' Wait for the socket to initialise
    Thread.Sleep(1000)

    Console.WriteLine("[CLIENT] Connecting to server...")

    ' Setup a dummy client
    Dim clientSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)
    Dim remoteEP As New IPEndPoint(IPAddress.Parse("127.0.0.1"), 8001)

    Try
        clientSock.Connect(remoteEP)

        ' Generate dummy file bytes to transfer
        Dim randomBytes(250000000) As Byte
        Dim randomizer As New RNGCryptoServiceProvider()
        randomizer.GetBytes(randomBytes)

        ' Append length at start of transmission
        Dim messageLength() As Byte = BitConverter.GetBytes(2499999996)
        randomBytes(0) = messageLength(0)
        randomBytes(1) = messageLength(1)
        randomBytes(2) = messageLength(2)
        randomBytes(3) = messageLength(3)

        clientSock.Send(randomBytes)
    Catch ex As Exception
        Console.WriteLine("[CLIENT] " + ex.ToString())
    End Try

```

```
While True
    Thread.Sleep(1000)
End While
End Sub
```

```
Private Sub Service()
    ' Setup server socket
    Dim hostSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)
    Dim endPoint As New IPEndPoint(IPAddress.Any, 8001)
    hostSock.Bind(endPoint)
    hostSock.Listen()

    serviceRunning = True
    Do While serviceRunning
        hostSock.BeginAccept(AddressOf ServiceAcceptCallback, hostSock)
    Loop
End Sub
```

```
Private Sub ServiceAcceptCallback(result As IAsyncResult)
    ' Accept async client and spawn an instance representing its state
    Console.WriteLine("[SERVER] Client accepted...")

    Dim clientSock As Socket = DirectCast(result.AsyncState, Socket).EndAccept(result)
    Dim client As New Client(clientSock, 1024)
    clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf
ServiceReceiveCallback), client)
End Sub
```

```
End Sub
```

```
Private Sub ServiceReceiveCallback(result As IAsyncResult)
    Dim client As Client = result.AsyncState
    Dim clientSock As Socket = client.socket
    Dim bytesReceived As Integer = clientSock.EndReceive(result)

    ' If there is data
    If bytesReceived > 0 Then
        client.bytesRead += bytesReceived ' Append the current bytes we have read

        ' Extract the length of the file we are downloading
        If Not client.isProcessing Then
            client.messageLength = BitConverter.ToUInt32(client.serviceBuffer, 0)
            client.isProcessing = True
            client.bytesRead -= 4
        End If

        ' Report the progress
        Dim progress As Integer = (client.bytesRead / client.messageLength) * 100
        Console.WriteLine("[SERVER] " + Convert.ToString(progress) + "% downloaded")
    End If
End Sub
```

```

' See if we have downloaded the entire file
If (client.bytesRead >= client.messageLength) Then
    client.isProcessing = False
    client.messageLength = 0
    client.bytesRead = 0
    Console.WriteLine("Download complete")
End If

```

```

' Get more data that is left in the pipe
clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf
ServiceReceiveCallback), client)
End Sub

```

End Module

This will simulate a client sending a ~250MB file across to a server, with the server parsing the header and assembling the fragments whilst calculating how much is left to go.

This should be enough to show you how to do it, keep in mind that the code above has no exception handling and can do with several more layers of abstraction. Definitely isn't reliable plug and play code.

elimagek - October 7, 2021, 6:07 am

(Oct 6, 2021 01:05 PM)Donkey Death Jam Wrote: [\[CITE\]](#) Here's a very crummy example I've whipped up:

```

Code Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Security.Cryptography
Imports System.Threading

```

Module Program

```

''' <summary>
''' Boolean representing whether the server is running/stopped
''' </summary>
Dim serviceRunning As Boolean = False

```

```

''' <summary>
''' Abstract representation of our client
''' </summary>
Private Structure Client
    ''' <summary>
    ''' Socket instance belonging to the client
    ''' </summary>
    Public socket As Socket

```

```

''' <summary>
''' Holds any received data from a client socket
''' </summary>
    Public serviceBuffer() As Byte

```

```

''' <summary>

```

''' Used to ensure correct content is being processed as a single message

''' </summary>

Public isProcessing As Boolean

''' <summary>

''' Length of the message we are processing

''' </summary>

Public messageLength As UInteger

''' <summary>

''' Bytes we've currently read

''' </summary>

Public bytesRead As UInteger

''' <summary>

''' Initialise the structure properties

''' </summary>

''' <param name="bufferSize">size of receive buffer </param>

Public Sub New(sock As Socket, bufferSize As Integer)

 socket = sock

 ReDim serviceBuffer(bufferSize)

 isProcessing = False

End Sub

End Structure

''' <summary>

''' Initial entry point

''' </summary>

''' <param name="args">Launch arguments</param>

Sub Main(args As String())

 Console.WriteLine("[SERVER] Initiating server...")

 ' Spin a server on an isolated thread

 Dim serviceWorker As New Thread(AddressOf Service)

 serviceWorker.IsBackground = True

 serviceWorker.Start()

 ' Wait for the socket to initialise

 Thread.Sleep(1000)

 Console.WriteLine("[CLIENT] Connecting to server...")

 ' Setup a dummy client

 Dim clientSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)

 Dim remoteEP As New IPEndPoint(IPAddress.Parse("127.0.0.1"), 8001)

 Try

 clientSock.Connect(remoteEP)

 ' Generate dummy file bytes to transfer

 Dim randomBytes(250000000) As Byte

```
Dim randomizer As New RNGCryptoServiceProvider()  
randomizer.GetBytes(randomBytes)
```

```
' Append length at start of transmission  
Dim messageLength() As Byte = BitConverter.GetBytes(249999996)  
randomBytes(0) = messageLength(0)  
randomBytes(1) = messageLength(1)  
randomBytes(2) = messageLength(2)  
randomBytes(3) = messageLength(3)
```

```
clientSock.Send(randomBytes)  
Catch ex As Exception  
    Console.WriteLine("[CLIENT] " + ex.ToString())  
End Try
```

```
While True  
    Thread.Sleep(1000)  
End While  
End Sub
```

```
Private Sub Service()  
    ' Setup server socket  
    Dim hostSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,  
ProtocolType.Tcp)  
    Dim endPoint As New IPEndPoint(IPAddress.Any, 8001)  
    hostSock.Bind(endPoint)  
    hostSock.Listen()  
  
    serviceRunning = True  
    Do While serviceRunning  
        hostSock.BeginAccept(AddressOf ServiceAcceptCallback, hostSock)  
    Loop
```

```
End Sub
```

```
Private Sub ServiceAcceptCallback(result As IAsyncResult)  
    ' Accept async client and spawn an instance representing its state  
    Console.WriteLine("[SERVER] Client accepted...")  
  
    Dim clientSock As Socket = DirectCast(result.AsyncState, Socket).EndAccept(result)  
    Dim client As New Client(clientSock, 1024)  
    clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf  
ServiceReceiveCallback), client)
```

```
End Sub
```

```
Private Sub ServiceReceiveCallback(result As IAsyncResult)  
    Dim client As Client = result.AsyncState  
    Dim clientSock As Socket = client.socket  
    Dim bytesReceived As Integer = clientSock.EndReceive(result)  
  
    ' If there is data
```

```

If bytesReceived > 0 Then
    client.bytesRead += bytesReceived ' Append the current bytes we have read

    ' Extract the length of the file we are downloading
    If Not client.isProcessing Then
        client.messageLength = BitConverter.ToUInt32(client.serviceBuffer, 0)
        client.isProcessing = True
        client.bytesRead -= 4
    End If

    ' Report the progress
    Dim progress As Integer = (client.bytesRead / client.messageLength) * 100
    Console.WriteLine("[SERVER] " + Convert.ToString(progress) + "% downloaded")
End If

' See if we have downloaded the entire file
If (client.bytesRead >= client.messageLength) Then
    client.isProcessing = False
    client.messageLength = 0
    client.bytesRead = 0
    Console.WriteLine("Download complete")
End If

' Get more data that is left in the pipe
clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf
ServiceReceiveCallback), client)
End Sub

```

End Module

This will simulate a client sending a ~250MB file across to a server, with the server parsing the header and assembling the fragments whilst calculating how much is left to go.

This should be enough to show you how to do it, keep in mind that the code above has no exception handling and can do with several more layers of abstraction. Definitely isn't reliable plug and play code.

thank you very much

not work with me

if you have sample source code with that i hope to upload it

(Oct 6, 2021 01:05 PM)Donkey Death Jam Wrote: [\[CITE\]](#) Here's a very crummy example I've whipped up:

```

Code Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Security.Cryptography
Imports System.Threading

```

Module Program

```

''' <summary>
''' Boolean representing whether the server is running/stopped

```

```
''' </summary>
```

```
Dim serviceRunning As Boolean = False
```

```
''' <summary>
```

```
''' Abstract representation of our client
```

```
''' </summary>
```

```
Private Structure Client
```

```
    ''' <summary>
```

```
    ''' Socket instance belonging to the client
```

```
    ''' </summary>
```

```
    Public socket As Socket
```

```
    ''' <summary>
```

```
    ''' Holds any received data from a client socket
```

```
    ''' </summary>
```

```
    Public serviceBuffer() As Byte
```

```
    ''' <summary>
```

```
    ''' Used to ensure correct content is being processed as a single message
```

```
    ''' </summary>
```

```
    Public isProcessing As Boolean
```

```
    ''' <summary>
```

```
    ''' Length of the message we are processing
```

```
    ''' </summary>
```

```
    Public messageLength As UInteger
```

```
    ''' <summary>
```

```
    ''' Bytes we've currently read
```

```
    ''' </summary>
```

```
    Public bytesRead As UInteger
```

```
    ''' <summary>
```

```
    ''' Initialise the structure properties
```

```
    ''' </summary>
```

```
    ''' <param name="bufferSize">size of receive buffer </param>
```

```
    Public Sub New(sock As Socket, bufferSize As Integer)
```

```
        socket = sock
```

```
        ReDim serviceBuffer(bufferSize)
```

```
        isProcessing = False
```

```
    End Sub
```

```
End Structure
```

```
''' <summary>
```

```
''' Initial entry point
```

```
''' </summary>
```

```
''' <param name="args">Launch arguments</param>
```

```
Sub Main(args As String())
```

```
    Console.WriteLine("[SERVER] Initiating server...")
```

```
    ' Spin a server on an isolated thread
```

```
    Dim serviceWorker As New Thread(AddressOf Service)
```

```

serviceWorker.IsBackground = True
serviceWorker.Start()

' Wait for the socket to initialise
Thread.Sleep(1000)

Console.WriteLine("[CLIENT] Connecting to server...")

' Setup a dummy client
Dim clientSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)
Dim remoteEP As New IPEndPoint(IPAddress.Parse("127.0.0.1"), 8001)

Try
    clientSock.Connect(remoteEP)

    ' Generate dummy file bytes to transfer
    Dim randomBytes(250000000) As Byte
    Dim randomizer As New RNGCryptoServiceProvider()
    randomizer.GetBytes(randomBytes)

    ' Append length at start of transmission
    Dim messageLength() As Byte = BitConverter.GetBytes(2499999996)
    randomBytes(0) = messageLength(0)
    randomBytes(1) = messageLength(1)
    randomBytes(2) = messageLength(2)
    randomBytes(3) = messageLength(3)

    clientSock.Send(randomBytes)
Catch ex As Exception
    Console.WriteLine("[CLIENT] " + ex.ToString())
End Try

While True
    Thread.Sleep(1000)
End While
End Sub

Private Sub Service()
    ' Setup server socket
    Dim hostSock As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)
    Dim endPoint As New IPEndPoint(IPAddress.Any, 8001)
    hostSock.Bind(endPoint)
    hostSock.Listen()

    serviceRunning = True
    Do While serviceRunning
        hostSock.BeginAccept(AddressOf ServiceAcceptCallback, hostSock)
    Loop

End Sub

```

```

Private Sub ServiceAcceptCallback(result As IAsyncResult)
    ' Accept async client and spawn an instance representing its state
    Console.WriteLine("[SERVER] Client accepted...")

    Dim clientSock As Socket = DirectCast(result.AsyncState, Socket).EndAccept(result)
    Dim client As New Client(clientSock, 1024)
    clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf
ServiceReceiveCallback), client)

End Sub

Private Sub ServiceReceiveCallback(result As IAsyncResult)
    Dim client As Client = result.AsyncState
    Dim clientSock As Socket = client.socket
    Dim bytesReceived As Integer = clientSock.EndReceive(result)

    ' If there is data
    If bytesReceived > 0 Then
        client.bytesRead += bytesReceived ' Append the current bytes we have read

        ' Extract the length of the file we are downloading
        If Not client.isProcessing Then
            client.messageLength = BitConverter.ToUInt32(client.serviceBuffer, 0)
            client.isProcessing = True
            client.bytesRead -= 4
        End If

        ' Report the progress
        Dim progress As Integer = (client.bytesRead / client.messageLength) * 100
        Console.WriteLine("[SERVER] " + Convert.ToString(progress) + "% downloaded")
    End If

    ' See if we have downloaded the entire file
    If (client.bytesRead >= client.messageLength) Then
        client.isProcessing = False
        client.messageLength = 0
        client.bytesRead = 0
        Console.WriteLine("Download complete")
    End If

    ' Get more data that is left in the pipe
    clientSock.BeginReceive(client.serviceBuffer, 0, 1024, 0, New AsyncCallback(AddressOf
ServiceReceiveCallback), client)
End Sub

```

End Module

This will simulate a client sending a ~250MB file across to a server, with the server parsing the header and assembling the fragments whilst calculating how much is left to go.

This should be enough to show you how to do it, keep in mind that the code above has no exception handling and can do with several more layers of abstraction. Definitely isn't reliable plug and play code.

i got source code but send file with error bytes

if file have 40 mb after sent is 237 mb

if you can fix this

[https://anonfiles\[WARNING POSSIBLY INFECTED\].com/Vcwbi6M0u5/File_Transfer_update_zip](https://anonfiles[WARNING POSSIBLY INFECTED].com/Vcwbi6M0u5/File_Transfer_update_zip)

PureCoder - October 7, 2021, 7:02 am

(Oct 7, 2021 06:07 AM)elImagek Wrote: [\[CITE\]](#) thank you very much

not work with me

if you have sample source code with that i hope to upload it

i got source code but send file with error bytes

if file have 40 mb after sent is 237 mb

if you can fix this

[https://anonfiles\[WARNING POSSIBLY INFECTED\].com/Vcwbi6M0u5/File_Transfer_update_zip](https://anonfiles[WARNING POSSIBLY INFECTED].com/Vcwbi6M0u5/File_Transfer_update_zip)

This is a good start.

Client.

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/synchronous-client-socket-example>

Server

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/synchronous-server-socket-example>

Better to read and learn. after that you can make your own progress bar thing.

elImagek - October 7, 2021, 7:05 am

(Oct 7, 2021 07:02 AM)PureCoder Wrote: [\[CITE\]](#) This is a good start.

Client.

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/synchronous-client-socket-example>

Server

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/synchronous-server-socket-example>

Better to read and learn. after that you can make your own progress bar thing.

if you can fix this only

[https://anonfiles\[WARNING POSSIBLY INFECTED\].com/Vcwbi6M0u5/File_Tr...update_zip](https://anonfiles[WARNING POSSIBLY INFECTED].com/Vcwbi6M0u5/File_Tr...update_zip)

#Kat - October 24, 2021, 2:50 pm

Dude apply the information given to your concept. Asking someone else to do the work for you is

just going to lead you right back here asking someone else to do another task for you. Simply put if you aren't ready for the basics don't jump into networking from the cliff ~no pun intended