Subject: TcpSocket, Know when distant connection have been closed
Posted by Xemuth on Sat, 12 Jun 2021 15:10:03 GMT
View Forum Message <> Reply to Message

Hello U++ !
I'm programming an application which rely on TcpSocket and Json sending & receiving.
A web server ask for data via a JSON command and I send him a formatted JSON. Here is how I do it :

```
while(!d_stopThread && !d_activeConnection.IsError() && d_activeConnection.IsOpen()){
 if(d_activeConnection.WaitRead()){
  Upp::String data;
  while(d_activeConnection.Peek()!= -1){
   data << char(d_activeConnection.Get());
  }

  if(data.GetCount() > 0){
   Upp::String sendingCmd = "";
   LLOG("[Server][Listener] Receiving  from web server: " + data.Left(20));
   sendingCmd = d_callbackServer(d_socket, data);
   if(sendingCmd.GetCount() > 0){
    LLOG("[Server][Listener] Sending to web server: " + sendingCmd.Left(20));
    d_activeConnection.Put(sendingCmd + "\n\0");
   }
  }
 }else{
  if(d_activeConnection.GetSOCKET() == -1)
   break;
  else
   Sleep(100);
 }
}
if(d_activeConnection.IsError()) LLOG("[Server][Listener] WebServer error: " +
d_socket.GetErrorDesc());
d_activeConnection.Clear();
LLOG("[Server][Listener] WebServer disconnected");
```

The web server send me data in JSON, I retrieve this data then I send back a json with the requested data. Here are a wireshark screenshot :

So here is my question, how to know when the distant socket have been closed ? I have trying many things but
either it's me who cuts the connection (probably because of a timeout or erro during WaitRead())
either I don't detect the FIN from the server side and dont react so my connection never reset

Subject: Re: TcpSocket receiving ghost data
Posted by Oblivion on Sat, 12 Jun 2021 15:34:33 GMT
View Forum Message <> Reply to Message

Hello Xemuth,

That's because that function is meant to convert pointer addresses into hexadecimal notation. What you see is a representation of the address of "data" String. What you need is HexEncode/HexDecode:

```
CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_COUT|LOG_FILE);

 String s = "A";
 RLOG(s);
 RLOG(FormatHex(~s));                  // Converts the pointer address to hex.
 RLOG(FormatHex(s.ToStd().c_str()));    // Converts the pointer address to hex.
 RLOG("---");
 RLOG(HexEncode(s));                   // Encodes the *content* of Strings into hex.
 RLOG(HexEncode(s.ToStd().c_str()));    // The same...
}
```

result:

```
A
7fff1bbab9a8
7fff1bbab958
---
41
41
```

Best regards,
Oblivion

Subject: Re: TcpSocket receiving ghost data
Posted by Xemuth on Sat, 12 Jun 2021 15:43:52 GMT
View Forum Message <> Reply to Message

Indeed, I read nothing, however, the boolean IsEOF() is still false, which mean their is still data to

read in my buffer. That's not valid ?

Subject: Re: TcpSocket receiving ghost data
Posted by Oblivion on Sat, 12 Jun 2021 16:00:48 GMT
View Forum Message <> Reply to Message

IsEof and isError do not include timeout errors - "it is not an error per se". (Hence: IsTimeout() :))

Upp::String data = d_activeConnection.GetLine();

What I can tell from the logs you shared, is, the GetLine() method is timing out.

My *guess* is that you need to add '\n' to the end of the data to be sent (That's how GetLine works. It will wait an EOL until it times out or encounters a socket error);

Best regards,
Oblivion

Subject: Re: TcpSocket receiving ghost data
Posted by Xemuth on Sat, 12 Jun 2021 16:47:53 GMT
View Forum Message <> Reply to Message

Thanks Oblivion,

In reality my problem seems to be a bit different from what I through it was, I'm updating the topic to make it stick more to my problem

Subject: Re: TcpSocket receiving ghost data
Posted by Oblivion on Sat, 12 Jun 2021 16:48:22 GMT
View Forum Message <> Reply to Message

One more thing (general recommendation for anyone reading this topic, not directed specifically at you):

You seem to have the assumption -or that's my impression- that GetLine() will always work on JSON formatted text.

If so, this is a false assumption. GetLine() is a convenience method for retrieving strictly formated texts which have line breaks. For this reason:

1) It cannot be effectively used with raw binary data, because it will always eat bytes '\r' (0xd) and '\n' (0xa).

2) Json (or XML) is not requred to have line breaks. That is optional and purely for cosmetics.
In fact line breaks in JSON/XML are not welcomed on networking circles because of the unnecessary overhead they add. (Think about it: A JSON with 1024 lines with linebreaks will add 1024 bytes to the document to be transferred)

3) My suggestion. Send the string length first thant retrieve the text, using TcpSocket::Get(length) method.

Best regards,
Oblivion

Subject: Re: TcpSocket receiving ghost data
Posted by Xemuth on Sat, 12 Jun 2021 17:23:00 GMT
View Forum Message <> Reply to Message

Thanks for recommendation Oblivion, working with socket without having knowledge of the thing is hard...

1 : I have changed the way I retrieve data (without using GetLine)

2 : The server with which I work is sending JSON data

3 : I thought about setting up this system with the web server developer

---

Subject: Re: TcpSocket receiving ghost data
Posted by Oblivion on Sat, 12 Jun 2021 18:03:30 GMT
View Forum Message <> Reply to Message

Hello Xemuth,

Quote:
So here is my question, how to know when the distant socket have been closed ? I have trying many things but
either it's me who cuts the connection (probably because of a timeout or erro during WaitRead())
either I don't detect the FIN from the server side and dont react so my connection never reset

Hard to say without knowing the transactions between S/C but I suspect that remote server may

have enabled SO_LINGER.
Otherwise, AFAIK, IsEof() or IsError() should return true, depending on the server.

See: https://man7.org/linux/man-pages/man7/socket.7.html

SO_LINGER
Sets or gets the SO_LINGER option. The argument is a linger structure.

```
struct linger {
    int l_onoff;    /* linger active */
    int l_linger;   /* how many seconds to linger for */
};
```

When enabled, a close(2) or shutdown(2) will not return until all queued messages for the socket have been successfully sent or the linger timeout has been reached. Otherwise, the call returns immediately and the closing is done in the background. When the socket is closed as part of exit(2), it always lingers in the background.

In general, the "polite" way in networking is to let the client close its connection to the server (after it is done with transactions.). But this is more a recommendation not a strict rule.

Best regards,
Oblivion

Subject: Re: TcpSocket receiving ghost data
Posted by Xemuth on Sun, 13 Jun 2021 21:06:46 GMT
View Forum Message <> Reply to Message

I found what I needed, in order to get noticed of pair closing socket (i.o client sending a FIN)

According to MSDN and the select(...) (from Winsock2) function :
Quote:For connection-oriented sockets, readability can also indicate that a request to close the socket has been received from the peer. If the virtual circuit was closed gracefully, and all data was received, then a recv will return immediately with zero bytes read. If the virtual circuit was reset, then a recv will complete immediately with an error code such as WSAECONNRESET. The presence of OOB data will be checked if the socket option SO_OOBINLINE has been enabled (see setsockopt).

To perform a correct client server ready to handle a FIN from a client, I did this :

Server.h

```
d_activeConnection.Timeout(500);
d_activeConnection.Linger(200);
SocketWaitEvent swe;
swe.Add(d_activeConnection, 0x7); // 0x7 is WAIT_ALL
while(!d_stopThread && !d_activeConnection.IsError() && d_activeConnection.IsOpen()){
 swe.Wait(500);
 if(swe[0] & WAIT_WRITE){
  if((swe[0] & WAIT_READ)){
   Upp::String data;

   int read =  d_activeConnection.Get(); //if read return -1 then pair have sent us FIN
   if(read == -1) break;
   data << char(read);
   do{ data << char(d_activeConnection.Get()); }while(d_activeConnection.Peek() != -1);

   if(data.GetCount() > 0){
    Upp::String sendingCmd = "";
    LLOG("[Server][Listener] Receiving  from web server: " + data.Left(20));
    sendingCmd = d_callbackServer(d_activeConnection, data);
    if(sendingCmd.GetCount() > 0){
     LLOG("[Server][Listener] Sending to web server: " + sendingCmd.Left(20));
     d_activeConnection.Put(sendingCmd);
    }
   }
  }
 }else{
  break;
 }
}
if(d_activeConnection.IsError()) LLOG("[Server][Listener] WebServer error: " +
d_activeConnection.GetErrorDesc());
d_activeConnection.Close();
LLOG("[Server][Listener] WebServer disconnected");
```