Subject: [SOLVED] TcpSocket Connect error
Posted by steffen on Tue, 11 Mar 2014 11:35:57 GMT
View Forum Message <> Reply to Message

Hi,

Newest uppsrc revision 7033.

Here on my Ubuntu 12.04 workstation I see a problem with TcpSocket.Connect, it always returns true, even if no server is listening at the other end:

```
TcpSocket client_socket;
client_socket.Timeout(5000);
if (client_socket.Connect("127.0.0.1", 12345)) // Connect to non existing server
{
 LOG("ERROR, Connect returned true");
}
else
{
 LOG("Correct, The connection could not be established");
}
```

The doc mentions the operation is blocking, but it is not. It does not say anything about the return value, I just assume it would be true on success.

It works if I comment the O_NONBLOCK from Socket.cpp->TcpSocket::SetupSocket(), like this:
if(fcntl(socket, F_SETFL, (fcntl(socket, F_GETFL, 0)/* | O_NONBLOCK*/))) {
I discovered this while trying to connect to a slow server and the not so reader friendly TcpSocket::RawConnect function passed on the SOCKERR(EINPROGRESS) check. No matter if a connection was established or not.

I don't have the in depth knowledge on the upp code base, to say if removing the O_NONBLOCK will cause undesired problems in other places.

Subject: Re: TcpSocket Connect error
Posted by mirek on Wed, 12 Mar 2014 09:52:47 GMT
View Forum Message <> Reply to Message

steffen wrote on Tue, 11 March 2014 07:35Hi,

Newest uppsrc revision 7033.

Here on my Ubuntu 12.04 workstation I see a problem with TcpSocket.Connect, it always returns true, even if no server is listening at the other end:

```
TcpSocket client_socket;
client_socket.Timeout(5000);
```

```
if (client_socket.Connect("127.0.0.1", 12345)) // Connect to non existing server
{
 LOG("ERROR, Connect returned true");
}
else
{
 LOG("Correct, The connection could not be established");
}
```
The doc mentions the operation is blocking, but it is not. It does not say anything about the return value, I just assume it would be true on success.

It works if I comment the O_NONBLOCK from Socket.cpp->TcpSocket::SetupSocket(), like this:
`if(fcntl(socket, F_SETFL, (fcntl(socket, F_GETFL, 0)/* | O_NONBLOCK*/))) {`
I discovered this while trying to connect to a slow server and the not so reader friendly TcpSocket::RawConnect function passed on the SOCKERR(EINPROGRESS) check. No matter if a connection was established or not.

I don't have the in depth knowledge on the upp code base, to say if removing the O_NONBLOCK will cause undesired problems in other places.

The documentation is misleading. It is only blocking w.r.t. DNS resolve.

Subject: Re: TcpSocket Connect error
Posted by steffen on Wed, 12 Mar 2014 16:24:40 GMT
View Forum Message <> Reply to Message

Thanks Mirek,

But what about the return value? I would expect TcpSocket.Connect to return true, when it has established a connection.
This is definitely not how it works, if the server is not responding.

I looked at the Linux man page on connect and for non blocking connections it says the following:
Quote:EINPROGRESS
The socket is nonblocking and the connection cannot be completed immediately.
It is possible to select(2) or poll(2) for completion by selecting the socket for writing.
After select(2) indicates writability, use getsockopt(2) to read the SO_ERROR option at level SOL_SOCKET to determine whether connect() completed successfully (SO_ERROR is zero) or unsuccessfully
(SO_ERROR is one of the usual error codes listed here, explaining the reason for the failure).

I have attached a small example showing the error and the solution mentioned in the man page:
When connecting to a non existing server, TcpSocket.Connect returns true, but fails on a subsequent write.

I have tested the example on both Windows and Linux, and both systems has the error.

So a solution could be one of these two:
1. Change TcpSocket::SetupSocket() to make a blocking socket:
Change Socket.cpp line 302 to
 if(fcntl(socket, F_SETFL, (fcntl(socket, F_GETFL, 0)))) {
or
2. Implement the getsockopt call to check the connection before returning from RawConnect.
I can make a patch to RawConnect if you would prefer that solution.

File Attachments

1) TcpSocketTest.7z, downloaded 471 times

Subject: Re: TcpSocket Connect error
Posted by mirek on Wed, 12 Mar 2014 18:07:20 GMT
View Forum Message <> Reply to Message

Well, this variant of Connect actually returns 'true' if host name was resolved, false if not.

It should be possible to make things blocking using WaitWrite after Connect, then error can be checked (Wait actually calls select).

In fact, I guess Connect should be changed to the call itself.

Mirek

Subject: Re: TcpSocket Connect error
Posted by steffen on Thu, 13 Mar 2014 07:28:04 GMT
View Forum Message <> Reply to Message

I used the WaitWrite in the Connect2 function in my example, but checking errno just after the WaitWrite returns, it still shows 115 (EINPROGRESS).
It only works if I call getsockopt afterwards, then optval is set to 111 (Connection refused).

I have improved the logging in my example, so now the output is like this:
Quote:Test1 Connecting
After Connect: 115 Operation now in progress
Connected
Sending: My Test String
ERROR: 111 Connection refused

Test2 Connecting

After WaitWrite: 115 Operation now in progress
optval: 111 Connection refused
SUCCESS: No Connection to non existing server.

Test1 is getting the error after the first write attempt, but it should have failed on connect.
Test2 calls getsockopt and detects the missing connection correctly.

File Attachments

1) TcpSocketTest.cpp, downloaded 486 times

Subject: Re: TcpSocket Connect error
Posted by steffen on Thu, 13 Mar 2014 07:43:42 GMT
View Forum Message <> Reply to Message

Here is a patched version of RawConnect, I have NOT tried it with IPv6:

```
bool TcpSocket::RawConnect(addrinfo *arp)
{
 if(!arp) {
  SetSockError("connect", -1, "not found");
  return false;
 }
 String err;
 for(int pass = 0; pass < 2; pass++) {
  addrinfo *rp = arp;
  while(rp)
  {
   if(rp->ai_family == AF_INET == !pass) // Try to connect IPv4 in the first pass
   {
    if (Open(rp->ai_family, rp->ai_socktype, rp->ai_protocol))
    {
     int r = connect(socket, rp->ai_addr, (int)rp->ai_addrlen);
     if(r != 0)
     {
      if (WaitWrite())
      {
       int optval = 0;
       socklen_t optlen = sizeof(optval);
       if (getsockopt(GetSOCKET(), SOL_SOCKET, SO_ERROR, (char*)&optval, &optlen) == 0)
       {
        if (optval == 0)
         r = 0;
        else
        {
         if(err.GetCount())
          err << '\n';
         err << TcpSocketErrorDesc(optval);
```

```
        }
       }
      }
     }
     if (r == 0)
     {
      mode = CONNECT;
      return true;
     }
     if(err.GetCount())
      err << '\n';
     err << TcpSocketErrorDesc(GetErrorCode());
     Close();
    }
   }
   rp = rp->ai_next;
  }
 }
 SetSockError("connect", -1, Nvl(err, "failed"));
 return false;
}
```

And the output from my connect test is:
Quote:Test1 Connecting
SUCCESS: No Connection to non existing server.

Test2 Connecting
SUCCESS: No Connection to non existing server.

Regards,
Steffen

Subject: Re: TcpSocket Connect error
Posted by mirek on Sun, 23 Mar 2014 16:16:51 GMT
View Forum Message <> Reply to Message

I am sorry for the delay, I had to think this through....

I am afraid I would not be happy with this change as proposed. The problem is that one of aims of non-blocking sockets was ability to provide the fully asynchronous mode of operation, as e.g. demonstrated in reference/GuiWebCrawler, where a multitude of HttpRequests is running concurently, in single thread.

By adding Wait into Connect in this exact way, this would no longer be possible, as Connect would start waiting for the connection (would be now blocking).

That said, I understand that current behaviour is not quite correct too, I just do not see the 'right'

solution here now....

Mirek

Subject: Re: TcpSocket Connect error
Posted by steffen on Mon, 24 Mar 2014 07:22:09 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thank you for the answer. I thought it might be complicated and I understand you concerns.
I have another suggestion then, that wont break anything:

Could you add a function that can be called prior to Connect, where one can determine if the socket should be blocking or not?
Storing a member variable containing either O_BLOCK or O_NONBLOCK and then use this variable in the SetupSocket function.

```
bool TcpSocket::SetupSocket()
{
#ifdef PLATFORM_WIN32
 connection_start = msecs();
 u_long arg = 1;
 if(ioctlsocket(socket, FIONBIO, &arg)) {
  SetSockError("ioctlsocket(FIO[N]BIO)");
  return false;
 }
#else
 #ifdef PLATFORM_BSD
  connection_start = msecs();
 #endif
 if(fcntl(socket, F_SETFL, (fcntl(socket, F_GETFL, 0) | O_NONBLOCK))) {
  SetSockError("fcntl(O_[NON]BLOCK)");
  return false;
 }
#endif
 return true;
}
```

Would be something like this:

```
bool TcpSocket::SetupSocket()
{
#ifdef PLATFORM_WIN32
 connection_start = msecs();
 u_long arg = 1;
 if(ioctlsocket(socket, mBlockingMode, &arg)) {
  SetSockError("ioctlsocket(FIO[N]BIO)");
  return false;
```

```
 }
#else
 #ifdef PLATFORM_BSD
  connection_start = msecs();
 #endif
 if(fcntl(socket, F_SETFL, (fcntl(socket, F_GETFL, 0) | mBlockingMode))) {
  SetSockError("fcntl(O_[NON]BLOCK)");
  return false;
 }
#endif
 return true;
}
```

As I wrote earlier, commenting out the O_NONBLOCK part also gives correct result.

Regards,
Steffen

Subject: Re: TcpSocket Connect error
Posted by mirek on Mon, 24 Mar 2014 09:21:31 GMT
View Forum Message <> Reply to Message

On the second thought, Connect with host/port is documented as blocking. Perhaps we could add waiting for write or just (temporary) switch to blocking socket just for this Connect variant?

Mirek

Subject: Re: TcpSocket Connect error
Posted by steffen on Mon, 24 Mar 2014 11:02:43 GMT
View Forum Message <> Reply to Message

Yes it would be correct according to documentation, but wouldn't it break the web crawler example?

It seems like I always end up struggling with the same parts when doing communication code. There is excellent helper functions for detecting read and write readiness. But I often have problems with the connection part, either because of slow lines or servers being offline.
It could be a great help if there was an IsConnected() function and/or a WaitConnect() function, like the WaitRead and WaitWrite functions.

Regards,
Steffen

Subject: Re: TcpSocket Connect error
Posted by mirek on Tue, 01 Apr 2014 17:51:27 GMT
View Forum Message <> Reply to Message

I have now added "WaitConnect", please check.

I am yet undecided whether to add WaitConnect at the end of Connect... for now, I have changed docs.

Subject: Re: TcpSocket Connect error
Posted by steffen on Wed, 02 Apr 2014 05:02:52 GMT
View Forum Message <> Reply to Message

Thank you very much Mirek, it works fine now.

Just leave it out of Connect. It's easy enough to do:

```
if (socket.Connect(...) && socket.WaitConnect())
{
 // Connected
}
else
{
 LOG(socket.GetErrorDesc());
}
```