TCP/IP Tutorial

Originally published in NL #115
by Alex Davies

Introduction

There are many programming systems intended for use with the internet (e.g., java, perl, cgi), but Liberty Basic is unfortunately not one of them. That doesn't need to stop you from harnessing the power of the internet in LB, though!

What is TCP/IP?

TCP/IP (Transfer Control Protocol / Internet Protocol) is the basic way that two computers communicate over the internet. It can be thought of as an instant messenger (e.g., MSN) conversation between two people. Each computer can send text to the other.

The usual way this is structured is as a client and server system. This can be thought of as the relationship between a diner and a waiter:

The diner will call the waiter.

>>The client computer connects to the server.

The waiter will ask what he can do for the diner.

>>The server computer sends a response.

The diner asks for a drink.

>>The client computer sends an instruction to the server.

The waiter brings the drink.

>>The server gives a response, for example the web page the client has requested.

The best way to understand this is through an example. The following is a typical http connection:

```
Client connects to www.libertybasic.com:80
Server accepts connection
Client sends: "GET resources.html"
Server sends: "< html >
        Liberty BASIC Resources - Bookmark this page!
        There's a lot of cool resources available for Liberty BASIC pr
ogrammers"
```

(And so on..)
I hope this demonstrates the nature of the internet. Its not magic, just a few clever techniques!

Introducing Mesock

MeSock.dll is a way which we can access the internet from LB without the limiting complexity we would otherwise have to wade through. The following functions use dll calls, but you don't need to understand them to use them!

You can get mesock.dll from:
http://oregonstate.edu/~reeset/html/other/programs/mesock32.dll

mesock32.dll

- Details
- Download
- 34 KB

Terry Reese: The code or software is provided free, in good faith, as-is, use at your own risk, etc. by the author, Terry Reese. By downloading and/or using the code you are agreeing to hold the author harmless from all effects and side-effects of using said code.

To begin with, your program will have to use the following command to prepare mesock for use:

```
    open "mesock32.dll" for dll as #me
```

Also, you must make sure that when your program closes, it closes mesock using the following command:

```
    close #me
```

Other than that all you need to do is paste the following four TCP functions at the end of your program.

```
''''Function TCPOpen()''''''''''
Function TCPOpen(address$,Port)
Timeout=1000
calldll #me, "Open", address$ As ptr,_
Port As Long,_
Timeout As Long, re As Long
TCPOpen=re
End Function

''''Function TCPReceive$()''''''''''
Function TCPReceive$(handle)
buffer=4096
all=0
calldll #me, "ReceiveA" ,handle As Long,_
buffer As Long,_
all As Long, re As long
if re<>0 then TCPReceive$ = winstring(re)
End Function

''''Function TCPPrint()''''''''''
Function TCPPrint(handle,text$)
calldll #me, "PrintA", handle As Long,_
text$ As ptr,re As Long
TCPPrint=re
End Function

''''Function TCPClose()''''''''''
Function TCPClose(handle)
calldll #me, "CloseA",handle As Long,_
TCPClose As Long
End Function
```

Once you have these vital components, you can make connections to internet servers at will.

Using The Four TCP Functions

TCPOpen

This must be called to connect to the server computer.

Parameters:

Address ($) - The Server's host address in either domain (www.libertybasic.com) or ip (192.34.65.1)
Port (numeric) - See Below]
Returns:

Handle (numeric) - The "handle" of the connection. This must be stored, as it is used in subsequent commands. 0 returned if failed.

TCPRecieve$

This will return any message that the server has sent back to your computer.

Parameters:

Handle (numeric) - The "handle" that was returned by TCPOpen.
Returns:

Message ($) - Any message sent by the server.

TCPPrint

This will send a message to the server.

Parameters:

Handle (numeric) - The "handle" that was returned by TCPOpen.
Text ($) - The text to send.
Returns:

Success (numeric) - -1 if sucessful

TCPClose

This disconnects from the server.

Parameters:

Handle (numeric) - The "handle" that was returned by TCPOpen.
Internet Protocols

These are ways which a client can request or send information to a server. It can be thought of as a mini language consisting of a few commands relevant to the purpose of the communication. The best known protocols are:

HTTP

HyperText Transfer Protocol

This is very quick and easy. There is only one major command, GET. The client asks for the file that it wants, and the server sends it. Its as simple as that.

Eg:

Connect to www.libertybasic.com:80
C: GET resources.html
S:
Liberty BASIC Resources - Bookmark this page!
There's a lot of cool resources available for Liberty BASIC programmers
(And so on..)

FTP

File Transfer Protocol

This is the alternative to HTTP, usually used for larger files. It is more complex, allowing for logging in, directory listing and other advanced functions.

SMTP

Simple Mail Transfer Protocol

This is the protocol by which email is sent. There are four basic commands, which must be used to send a successful email: HELO; MAIL FROM; RCPT TO; DATA.

Eg:

```
Connect to smtp.myisp.com:25
S: 220 mail18.svr.pol.co.uk ESMTP Exim 4.14 Fri, 07 Nov 2003 19:52:30 +0000
C: HELO myisp.com
S: 250 mail18.svr.pol.co.uk Hello modem-2864.buffalo.dialup.pol.co.uk [217.134.75.48]
C: MAIL FROM: TBI@hotmail.com >
S: 250 OK
C: RCPT TO: TBI@arro.org.uk >
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
C: SUBJECT: Read this email - I am the subject!
C: Hello,
C: I am the body of the message.
C:
C: TheBlazingIcicle
C: .
S: 250 OK id=1AICel-00079h-Cc
```

The SMTP protocol is fairly simple, and you just need to substitute the information you want into the "conversation" above.

My example program below is a simple smtp client.

POP

Post Office Protocol
This is the protocol used to collect emails from a server. It is also simple, but I will never finish this tutorial if I explain every protocol there is!

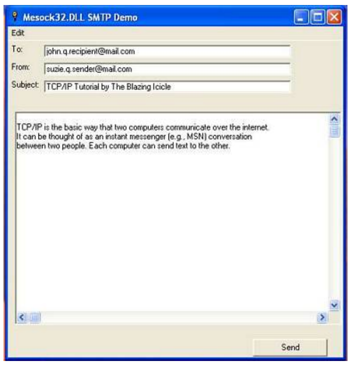
Ports

You may have noticed that the familiar server name in the previous examples (eg www.libertybasic.com) is suffixed by a number (eg :80). This number is the port of the server. A server can have many ports. The important thing for us is that each protocol has a port that is usually associated with it. (eg http:80 smtp:25) It is normal to write the port after a colon after the server's name.

You must specify which port you want to connect to when you make a connection.

Demo

TCP/IP Example: A Simple Mail Client
Mail client made with Mesock.dll



To: john.q.recipient@mail.com
From: suzie.q.sender@mail.com
Subject: TCP/IP Tutorial by The Blazing Icicle

TCP/IP is the basic way that two computers communicate over the internet.
It can be thought of as an instant messenger (e.g., MSN) conversation
between two people. Each computer can send text to the other.

```
'simple smtp client
WindowWidth = 480 : WindowHeight = 500
open "mesock32.dll" for dll as #me

statictext  #main, "To:", 5,5,40,20
statictext  #main, "From:", 5,30,40,20
statictext  #main, "Subject:", 5,55,40,20
button      #main.default, "Send",[send],UL, 345, 420, 105, 25
textbox     #main.to, 50, 5, 350, 20
textbox     #main.fr, 50, 30, 350, 20
textbox     #main.su, 50, 55, 350, 20
texteditor  #main.te, 10, 100, 460, 300

Open "Mesock32.DLL SMTP Demo" for window as #main
print #main, "trapclose [quit]"

wait

[quit]
close #main
close #me
end

[send]
#main.to "!contents? t0$"
#main.fr "!contents? fr$"
#main.su "!contents? su$"
#main.te "!contents? body$"
at=instr(t0$,"@")
```

```
server$=right$(t0$,len(t0$)-at)
print "Connecting To Server:"; server$
th = TCPOpen(server$, 25)
print "TCP Handle Opened: ";th
recvd$=TCPReceive$(th)
print "GET: ";recvd$
txt$="HELO "+server$
gosub [put]
txt$="MAIL FROM:<"+fr$+">"
gosub [put]
txt$="RCPT TO:<"+t0$+">"
gosub [put]
txt$="DATA"
gosub [put]
txt$="SUBJECT: "+su$
gosub [put]
txt$=body$
gosub [put]
txt$="."
gosub [put]
wait

[put]
lret = TCPPrint(th, txt$)
print "PUT: ";txt$
recvd$=TCPReceive$(th)
print "GET: ";recvd$
return

''''Function TCPOpen()'''''''''
Function TCPOpen(address$,Port)
Timeout=1000
calldll #me, "Open", address$ As ptr,_
Port As Long,_
Timeout As Long, re As Long
TCPOpen=re
End Function

''''Function TCPReceive$()'''''''''
Function TCPReceive$(handle)
buffer=4096
all=0
calldll #me, "ReceiveA" ,handle As Long,_
buffer As Long,_
all As Long, re As long
if re<>0 then TCPReceive$ = winstring(re)
```

```
End Function

''''Function TCPPrint()''''''''''
Function TCPPrint(handle,text$)
calldll #me, "PrintA", handle As Long,_
text$ As ptr,re As Long
TCPPrint=re
End Function

''''Function TCPClose()''''''''''
Function TCPClose(handle)
calldll #me, "CloseA",handle As Long,_
TCPClose As Long
End Function
```

TCP/IP Tutorial | Introduction | What is TCP/IP? | Introducing Mesock | Using The Four TCP Functions | TCPOpen | TCPRecieve$ | TCPPrint | TCPClose | HTTP | FTP | SMTP | POP | Ports | Demo