

Chapter 22: Connecting with a Network

This chapter discusses how to use the BASIC-256 networking statements. Networking in BASIC-256 will allow for a simple "socket" connection using TCP (Transmission Control Protocol). This chapter is not meant to be a full introduction to TCP/IP socket programming.

Socket Connection:

TCP stream sockets create a connection between two computers or programs. Packets of information may be sent and received in a bi-directional (or two way) manner over the connection.

To start a connection we need one computer or program to act as a server (to wait for the incoming telephone call) and the other to be a client (to make the telephone call). Illustration 40 shows graphically how a stream connection is made.

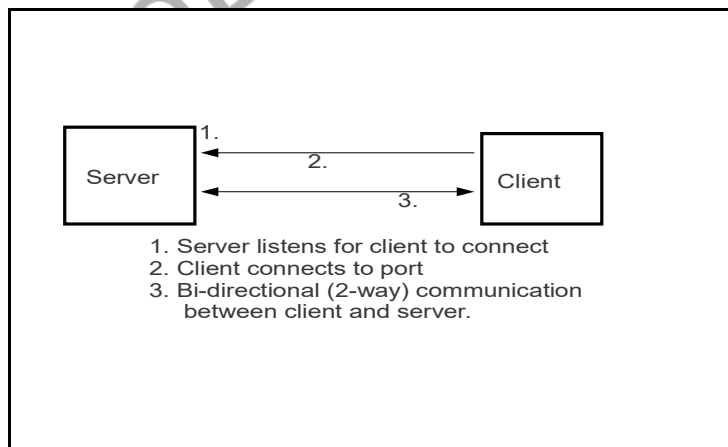


Illustration 40: Socket Communication

Just like with a telephone call, the person making the call (client) needs to know the phone number of the person they are calling (server). We call that number an IP address. BASIC-256 uses IP version 4 addresses that are usually expressed as four numbers separated by periods (A.B.C.D) where A, B, C, and D are integer values from 0 to 255.

In addition to having the IP address for the server, the client and server must also talk to each-other over a port. You can think of the port as a telephone extension in a large company. A person is assigned an extension (port) to answer (server) and if you want to talk to that person you (client) call that extension.

The port number may be between 0 and 65535 but various Internet and other applications have been reserved ports in the range of 0-1023. It is recommended that you avoid using these ports.

A Simple Server and Client:

```
1 # simpleserver.kbs
2 # send a message to the client on port 999
3
4 print "listening to port 9999 on " + netaddress()
5 NetListen 9999
6 NetWrite "The simple server sent this message."
7 NetClose
```

Program 131: Simple Network Server

```
1 # simpleclient.kbs
2 # connect to simple server and get the message
3 #
4 input "What is the address of the simple_server?",
  addr
5 if addr = "" then addr = "127.0.0.1"
6 #
```

```

7 NetConnect addr, 9999
8 print NetRead
9 NetClose

```

Program 132: Simple Network Client

```
listening to port 9999 on xx.xx.xx.xx
```


Sample Output 131: Simple Network Server


```


What is the address of the simple_server?
The simple server sent this message.


```


Sample Output 132: Simple Network Client


 <p>New Concept</p>	<pre> netaddress netaddress () </pre> <p>Function that returns a string containing the numeric IPv4 network address for this machine.</p>
--	--

 <p>New Concept</p>	<pre> netlisten portnumber netlisten (portnumber) netlisten socketnumber, portnumber netlisten (socketnumber, portnumber) </pre> <p>Open up a network connection (server) on a specific port address and wait for another program to connect. If <i>socketnumber</i> is not specified socket number zero (0) will be used.</p>
---	--

 New Concept	<pre>netclose netclose () netclose socketnumber netclose (socketnumber)</pre> <p>Close the specified network connection (socket). If <i>socketnumber</i> is not specified socket number zero (0) will be closed.</p>
---	---

 New Concept	<pre>netwrite string netwrite (string) netwrite socketnumber, string netwrite (socketnumber, string)</pre> <p>Send a string to the specified open network connection. If <i>socketnumber</i> is not specified socket number zero (0) will be written to.</p>
---	--

 New Concept	<pre>netconnect servername, portnumber netconnect (servername, portnumber) netconnect socketnumber, servername, portnumber netconnect (socketnumber, servername, portnumber)</pre> <p>Open a network connection (client) to a server. The IP address or host name of a server are specified in the <i>servername</i> argument, and the specific network port number. If <i>socketnumber</i> is not specified socket number zero (0) will be used for the connection.</p>
---	--

 <p>New Concept</p>	<pre>netread netread () netread (<i>socketnumber</i>)</pre> <p>Read data from the specified network connection and return it as a string. This function is blocking (it will wait until data is received). If <i>socketnumber</i> is not specified socket number zero (0) will be read from.</p>
---	---

Network Chat:

This example adds one new function (**netdata**) to the networking statements we have already introduced. Use of this new function will allow our network clients to process other events, like keystrokes, and then read network data only when there is data to be read.

The network chat program (Error: Reference source not found) combines the client and server program into one. If you start the application and it is unable to connect to a server the error is trapped and the program then becomes a server. This is one of many possible methods to allow a single program to fill both roles.

```
1 # chat.kbs
2 # use port 9999 for simple chat
3
4 input "Chat to address (return for server or local
   host)?", addr
5 if addr = "" then addr = "127.0.0.1"
6 #
7 # try to connect to server - if there is not one
   become one
8 try
9     NetConnect addr, 9999
10 catch
```

```
11     print "starting server - waiting for chat client"
12     NetListen 9999
13 end try
14 print "connected"
15
16 while true
17     # get key pressed and send it
18     k = key
19     if k <> 0 then
20         call show(k)
21         netwrite string(k)
22     end if
23     # get key from network and show it
24     if NetData() then
25         k = int(NetRead())
26         call show(k)
27     end if
28     pause .01
29 end while
30 end
31
32 subroutine show(keyvalue)
33     if keyvalue=16777220 then
34         print
35     else
36         print chr(keyvalue);
37     end if
38 end subroutine
```

Program 133: Network Chat

The following is observed when the user on the client types the message "HI SERVER" and then the user on the server types "HI CLIENT".


```
Chat to address (return for server or local
host)?
starting server - waiting for chat client
```


```
connected
HI SERVER
HI CLIENT
```

Sample Output 133.1: Network Chat (Server)

```
Chat to address (return for server or local
host)?
connected
HI SERVER
HI CLIENT
```

Sample Output 133.2: Network Chat (Client)

 <p>New Concept</p>	<pre>netdata or netdata() netdata (socketnumbr)</pre> <p>Returns true if there is network data waiting to be read. This allows for the program to continue operations without waiting for a network packet to arrive.</p>
---	---

 <p>Big Program</p>	<p>The big program this chapter creates a two player networked tank battle game. Each player is the white tank on their screen and the other player is the black tank. Use the arrow keys to rotate and move. Shoot with the space bar.</p>
---	---

```
1 # battle.kbs
```

```
2 # uses port 9998 for server
3
4 spritedim 4
5 call tanksprite(0,white) # me
6 call tanksprite(1,black) # opponent
7 call projectilesprite(2,blue) # my shot
8 call projectilesprite(3,red) # opponent shot
9
10 kspace = 32
11 kleft = 16777234
12 kright = 16777236
13 kup = 16777235
14 kdown = 16777237
15
16 dr = pi / 20 # direction change
17 dxy = 2.5 # move speed
18 shotdxy = 5 # shot move speed
19 port = 9998 # port to communicate on
20
21 print "Tank Battle - You are the white tank."
22 print "Your mission is to shoot and kill the"
23 print "black one. Use arrows to move and"
24 print "space to shoot."
25 print
26
27 input "Are you the server? (y or n)", mode
28 if mode = "y" then
29     print "You are the server. Waiting for a client to
    connect."
30     NetListen port
31 else
32     input "Server Address to connect to (return for
    local host)?", addr
33     if addr = "" then addr = "127.0.0.1"
34     NetConnect addr, port
35 end if
36
37 # set my default position and send to my opponent
38 x = 100
```



```
39 y = 100
40 r = 0
41 # projectile position direction and visible
42 p = false
43 px = 0
44 py = 0
45 pr = 0
46 call writeposition(x,y,r,p,px,py,pr)
47
48
49 # update the screen
50 color green
51 rect 0, 0, graphwidth, graphheight
52 spriteshow 0
53 spriteshow 1
54 spriteplace 0, x, y, 1, r
55 while true
56     # get key pressed and move tank on the screen
57     k = key
58     if k <> 0 then
59         if k = kup then
60             x = ( graphwidth + x + sin(r) * dxy ) %
graphwidth
61             y = ( graphheight + y - cos(r) * dxy ) %
graphheight
62         end if
63         if k = kdown then
64             x = ( graphwidth + x - sin(r) * dxy ) %
graphwidth
65             y = ( graphheight + y + cos(r) * dxy ) %
graphheight
66         end if
67         if k = kleft then r = r - dr
68         if k = kright then r = r + dr
69         if k = kspace then
70             pr = r
71             px = x
72             py = y
73             p = true
```

```
74         spriteshow 2
75     end if
76     spriteplace 0, x, y, 1, r
77     call writeposition( x, y, r, p, px, py, pr )
78     if spritecollide( 0, 1 ) then
79         netwrite "F"
80         print "You just ran into the other tank and
you both died. Game Over."
81     end
82     end if
83 end if
84 # move my projectile (if there is one)
85 if p then
86     px = px + sin( pr ) * shotdxy
87     py = py - cos( pr ) * shotdxy
88     spriteplace 2, px, py, 1, pr
89     if spritecollide( 1, 2 ) then
90         NetWrite "W"
91         print "You killed your opponent. Game over."
92     end
93     end if
94     if px < 0 or px > graphwidth or py < 0 or py >
graphheight then
95         p = false
96         spritehide 2
97     end if
98     call writeposition( x, y, r, p, px, py, pr )
99 end if
100 #
101 # get position from network and
102 # set location variables for the opponent
103 # flip the coordinates as we decode
104 while NetData()
105     position = NetRead()
106     while position <> ""
107         if left(position,1) = "W" then
108             print "You Died. - Game Over"
109         end
110     end if
```

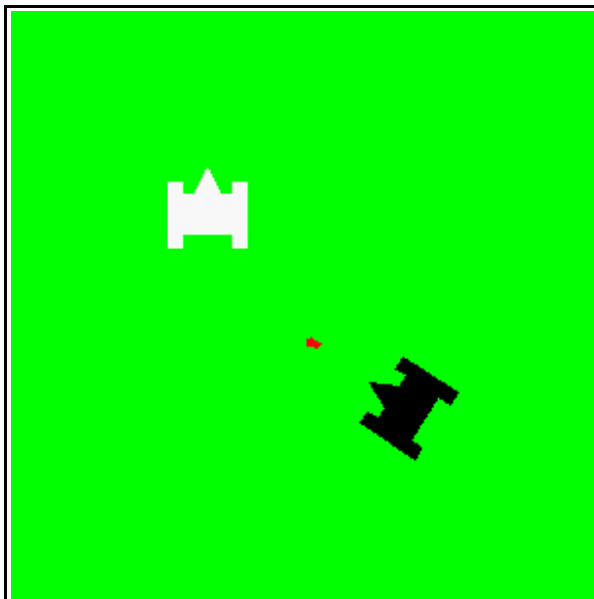
```

111         if left(position,1) = "F" then
112             print "You were hit and you both died. -
Game Over"
113         end
114     end if
115     op_x = graphwidth - unpad( ref( position ),
3)
116     op_y = graphheight - unpad( ref( position ),
3)
117     op_r = pi + unpad( ref( position ), 5)
118     op_p = unpad( ref( position ), 1)
119     op_px = graphwidth - unpad( ref( position ),
3)
120     op_py = graphheight -
unpad( ref( position ), 3)
121     op_pr = pi + unpad( ref( position ), 5)
122     # display opponent
123     spriteplace 1, op_x, op_y, 1, op_r
124     if op_p then
125         spriteshow 3
126         spriteplace 3, op_px, op_py, 1, op_pr
127     else
128         spritehide 3
129     end if
130     end while
131 end while
132 #
133 pause .05
134 end while
135
136 subroutine writeposition(x,y,r,p,px,py,pr)
137     position = lpad( int( x ), 3 ) + lpad( int( y ),
3 ) + lpad( r, 5 ) + lpad( p, 1 ) + lpad( int( px ),
3 ) + lpad( int( py ), 3 ) + lpad( pr, 5 )
138     NetWrite position
139 end subroutine
140
141 function lpad( n, l )
142     # return a number left padded in spaces

```


```
143     s = left( n, 1 )
144     while length( s ) < 1
145         s = " " + s
146     end while
147     return s
148 end function
149
150 function unpad( ref( l ), l )
151     # return a number at the begining padded in l
    spaces
152     # and shorten the string by l that we just pulled
    off
153     n = float( left( l, l ) )
154     if length( l ) > 1 then
155         l = mid( l, l + 1, 99999 )
156     else
157         l = ""
158     end if
159     return n
160 end function
161
162 subroutine tanksprite( spritenumber , c )
163     color c
164     spritepoly spritenumber, {0,0, 7,0, 7,7, 14,7,
    20,0, 26,7, 33,7, 33,0, 40,0, 40,40, 33,40, 33,33,
    7,33, 7,40, 0,40}
165 end subroutine
166
167 subroutine projectilesprite( spritenumber, c)
168     color c
169     spritepoly spritenumber, {3,0, 3,8, 0,8}
170 end subroutine
```

Program 134: Network Tank Battle



Sample Output 45: Adding Machine - Using Exit While

Exercises:

 <p>Word Search</p>	<pre> m r d t n s i p n n j r f d o c k e e e v v r c l r t g s t p h k i o s d e o c k e e w i a r k l o t n t l e v v c c n t e t r e t t n t n n e t r c x g o e e n e x p o r t m n c n e t i r w t e n t </pre> <p>client, listen, netclose, netconnect, netlisten, netread, network, netwrite, port, server, socket, tcp</p>
--	--



Problems

1. Modify Problem 4 from the keyboard control chapter to create a network client/server 2 player "ping-pong" game.
2. Write a simple server/client rock-paper-scissors game where two players will compete.
3. Write a complex network chat server that can connect to several clients at once. You will need a server process to assign each client a different port on the server for the actual chat traffic.

Free eBook Edition